NPS-ME-99-005

NAVAL POSTGRADUATE SCHOOL
Monterey, California

A Graphic User Interface (GUI) for Generating NPS Autonomous Underwater Vehicle (AUV) Execution Script Files

by

Joël Dolèac

August 1999

Approved for public release; distribution is unlimited.

Prepared for: Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM Robert C. Chaplin
Superintendent

R. Elster
Provost

This report was prepared for the Center for Autonomous Underwater Vehicle Research (CAUVR) and funded by the Office of Naval Research (ONR) (Dr. Tom Curtin) under Project N° 000 1498 WR 30175.

This report was prepared by:

Joel G. Doléac
French ENIT Student

Reviewed by:

for Terry R. Mac Nelley
Department of Mechanical Engineering

Released by:

D. W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1999	Project Report

4. TITLE AND SUBTITLE
A GRAPHIC USER INTERFACE (GUI) FOR GENERATING NPS AUTONOMOUS UNDERWATER VEHICLE (AUV) EXECUTION SCRIPT FILES

5. FUNDING NUMBERS
000 1498 WR 30175

6. AUTHOR
Joël G. Doléac

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Mechanical Engineering Department
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
800 North Quincy St.
Arlington, VA 22217

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The Naval Postgraduate School is on the leading edge of Autonomous Underwater Vehicle (AUV) research and has developed its own AUV called Phoenix. A new AUV is being constructed in order to increase the vehicle's mission capabilities. Despite its autonomy, the AUV needs to have the mission plan defined before starting. For that purpose, a text file containing a mission script is sent to the computer inside the robot. However, the file syntax is very precise and, thus, this approach to generating missions can be unsafe because it is easy to include typing errors in the text file. This project explains the creation of a Graphic User Interface (GUI) used to automatically generate the text file. Visual Prolog is used to realize such an interface, offering capabilities to simplify the writing of the Prolog code. The results of this work is a GUI that eliminates typing incorrect commands, provides a visualization of the commanded trajectory, and checks the syntax of the text. In this way, users can easily, quickly and safely build the mission plan.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Autonomous Underwater Vehicles, Graphic User Interface, Mission script	154
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

ABSTRACT

The Naval Postgraduate School is on the leading edge of Autonomous Underwater Vehicle (AUV) research and has developed its own AUV called Phoenix. A new AUV is being constructed in order to increase the vehicle's mission capabilities.

Despite its autonomy, the AUV needs to have the mission plan defined before starting. For that purpose, a text file containing a mission script is sent to the computer inside the robot. However, the file syntax is very precise and, thus, this approach to generating missions can be unsafe because it is easy to include typing errors in the text file.

This project explains the creation of a Graphic User Interface (GUI) used to automatically generate the text file. Visual Prolog is used to realize such an interface, offering capabilities to simplify the writing of the Prolog code.

The results of this work is a GUI that eliminates typing incorrect commands, provides a visualization of the commanded trajectory, and checks the syntax of the text. In this way, users can easily, quickly and safely build the mission plan.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

AUV	Autonomous Underwater Vehicle
A/D	Analog to Digital
DGPS	Differential Global Positioning System
D/A	Digital to Analog
GPS	Global Positioning System
GUI	Graphic User Interface
NPS	Naval Postgraduate School
RBM	Rational Behavior Model

ACKNOWLEDGEMENTS

First of all, I would like to acknowledge Professor Tony Healey who welcomed us to the AUV research group and give me support and guidance throughout this project.

I also wish to thank Dr. Don Brutzman for his support, his advice and his enthusiasm. I am also very grateful to Dr. David Marco who provided as much help as I needed and whom support, guidance and knowledge have been indispensable for the outcome of my work.

I would like to thank Didier Léandri and Fatima Benjou, who offered me the opportunity to live this great experience abroad.

A special thanks to Gwladys Piton, Samuel Lalaque and Sébastien Garibal, three other French students for the support they provided to me.

I wish to recognize partial support for this project for the Office of Naval Research (ONR) (Dr. Tom Curtin) under Project N° 000 1498 WR 30175.

I. INTRODUCTION

A. BACKGROUND

The Naval Postgraduate School is very involved in Autonomous Underwater Vehicle (AUV) research, and has therefore created a Center for Autonomous Underwater Vehicle Research (CAUVR) that is exploring many concepts in the design and control of AUVs. An Autonomous Underwater Vehicle is a self-contained unmanned vehicle used for missions such as surveying, pollution detection or mine countermeasure in shallow waters.

B. MOTIVATION AND GOALS

The NPS AUV uses a text file to define the mission plan before each experiment or mission. When this file is interpreted by the robot, its sequences robot control commands. The text file simply provides the different steps to follow.

This report outlines the development of a Graphic User Interface (GUI). The purpose of this GUI is the automatic generation of the mission-script text file. Thus, this approach makes it easier and faster for human operators to generate safe mission files.

C. SUMMARY OF CHAPTERS

The present chapter describes to the background, motivations and goals for this project. Chapter II presents related work for the NPS AUV project, the Virtual World used to accelerate its development, and previous work on a GUI. Chapter III states the explanation of the problem and the interest of a GUI. Chapter IV explains Prolog language and its logic. Chapter V shows the different tools of Visual Prolog that help in the creation of a GUI. Chapter VI outlines rationale followed in the creation of the code and provides a simple explanation of Prolog use. Chapter VII explains the way to use the created GUI. Finally, Chapter VIII contains the conclusions and recommendations for future work.

II. RELATED WORK

A. INTRODUCTION

Research on Autonomous Underwater Vehicles has been an ongoing project at the Naval Postgraduate School of Monterey (NPS) since 1987 through the *Phoenix* project [Healey 90,92] [Brutzman 96].

This section provides a general overview of the Phoenix vehicle (hardware and software) and the differences that will appear on the new AUV. It also includes a presentation of the virtual world used to predict the AUV reaction in particular environments and conditions, as well as a presentation of an previous work on a Graphic User Interface (GUI) [Davis, 96].

B. THE AUV PHOENIX

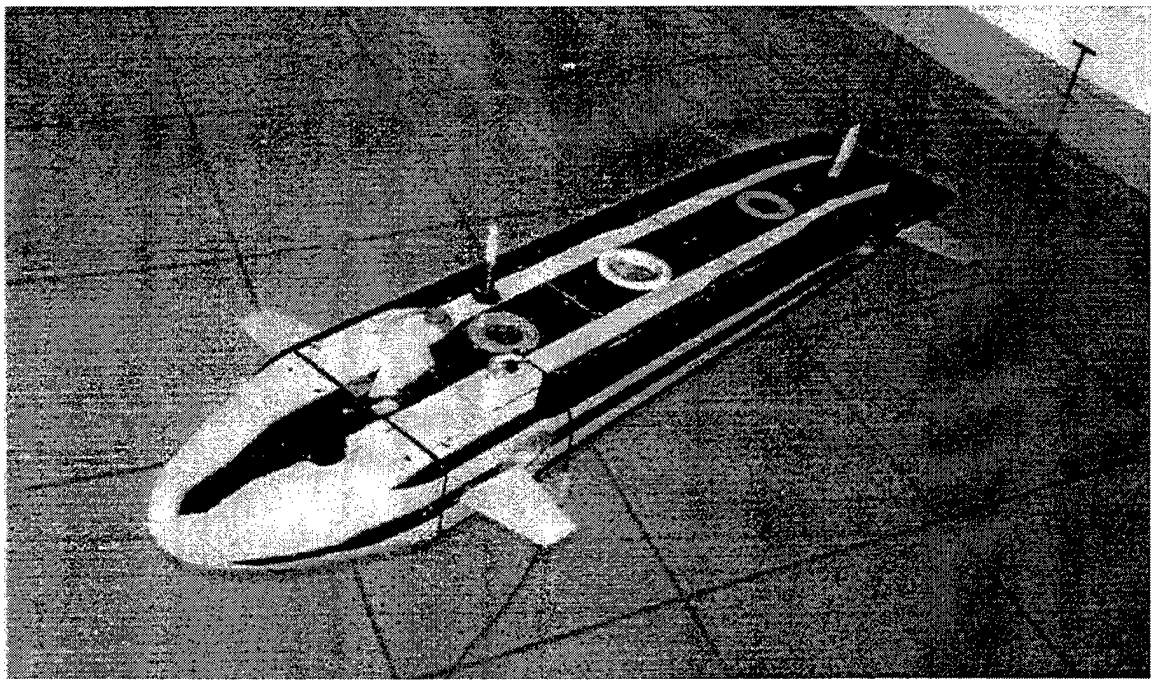

Figure II.1: Phoenix AUV undergoing testing at the Center for AUV Research (CAUVR) laboratory test tank in early 1995.

1. Physical Description

The Naval Postgraduate School *Phoenix* AUV is a complex robot, which contains various motors, controllers, servo-amplifiers and computers in a watertight hull. A internal view of the hardware layout is shown below in the figure II.2.

The AUV is approximately 2.4 meter long, 0.46 meter wide and 0.31 meter deep. It has a 2 psi pressurized aluminum hull with a free-flooding nose cone that houses some of the AUV's measurement devices. The vehicle is designed to be neutrally buoyant at three hundred and eighty seven pounds with a designed depth at twenty feet. It can be launched either from shore or from a boat. Lead acid batteries providing endurance up to two hours electrically power the submarine.

Two computers provide the control of the devices. These two computers can easily communicate together via an internal Ethernet network. The Ethernet can provide Internet connectivity to the boat through a tether. This tether can be used to monitor each process, collect data, or to intervene with an operational fault occurs. Ordinarily, the tether is only used when the AUV is being tested on shore or downloading test data at sea.

For the survey and mine countermeasure purposes mentioned above, several devices have been installed in the AUV. Some are intended for navigation and others are used for measurements. The following list details the primary pieces of hardware and their purposes:

- Four sonars:
  - ✓ Doppler sonar for the speed over the ground (RDI),
  - ✓ Sontek ADV for water particle relative velocity,
  - ✓ Obstacle detection (ST 725 model),
  - ✓ Obstacle classification (ST 1000 model),
- GPS and DGPS for tracking the vehicle latitude and longitude,
- Dive Tracker short baseline sonar navigation for precision tracking,
- Gyros for sensing the vehicle's orientation by measuring angles and rates for roll, pitch and yaw respectively. They will be also put out in the new boat.
- A pressure sensitive depth cell.

NAVAL POSTGRADUATE SCHOOL
CENTER FOR AUV RESEARCH

PHOENIX AUV FOR MINE RECONNAISSANCE/
NEUTRALIZATION IN VERY SHALLOW WATERS

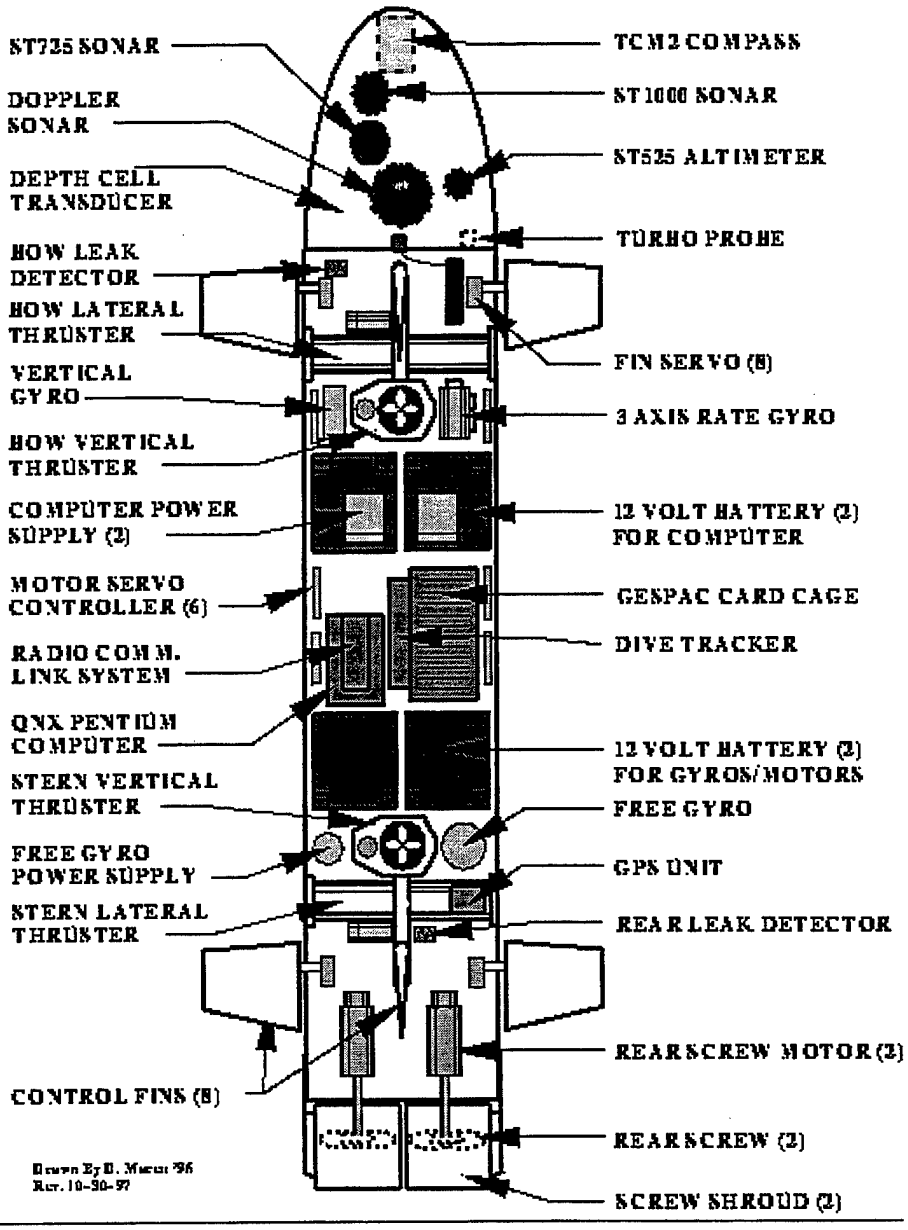

Figure II.2: The components inside the Phoenix AUV.

2. Software

The Phœnix AUV uses a tri-level software architecture called the Rational Behavior Model (RBM). RBM divides responsibilities into areas of open-ended strategic planning, soft-real-time tactical analysis, and hard real time execution level control. The RBM architecture has been created as a model of a manned submarine operational structure. The correspondence between the three levels and a submarine crew is shown in the Figure II.3.

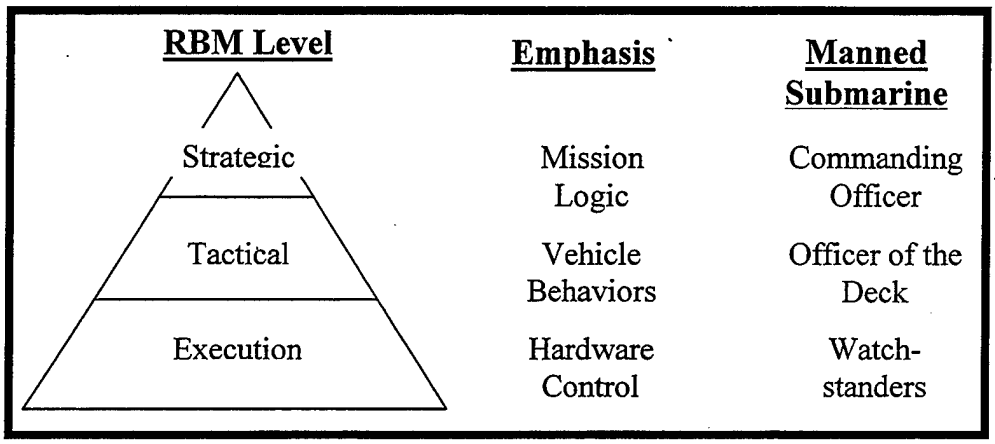

Figure II.3: Relational Behavior Model tri-level architecture hierarchy with level emphasis and submarine equivalent listed [Holden 95].

The **Execution Level** assures the interface between hardware and software. Its tasks are to underlay the stability of the vehicle, to control the individual devices, and to provide data to the tactical level.

The **Tactical Level** provides a software level that interfaces with both the Execution level and the Strategic level. Its chores are to give to the Strategic level indications of vehicle state, completed tasks and execution level commands. The Tactical level selects the tasks needed to reach the goal imposed by the Strategic level. It operates in terms of discrete events.

The **Strategic Level** controls the completion of the mission goals. The mission specifications are inside this level.

C. DESIGN OF THE NEW AUV

During its planned missions like bottom surveying or mine hunting, the AUV needs to have the ability to take and keep its position in a dynamic environment relative to a local stationary object. This ability, through the use of sensors and actuators (propellors, fins, thrusters), consumes power. The power capacity is very important in an AUV because it will determine the duration of the mission. In order to increase the range of the boat, a new NPS AUV is being manufactured.

This new boat is very similar to Phœnix. Actually the global shape for both hardware and software has been maintained. The main differences stand in the addition of two ballast chambers (lengthening the hull) and the increase of the power capacity. The new vehicle will use a 48 volt batteries pack instead of a 24 volt batteries pack. The goal of the ballast chamber is to enable the AUV to set on the ocean's bottom in a mechanical way (making it heavier) without consuming a lot of power.

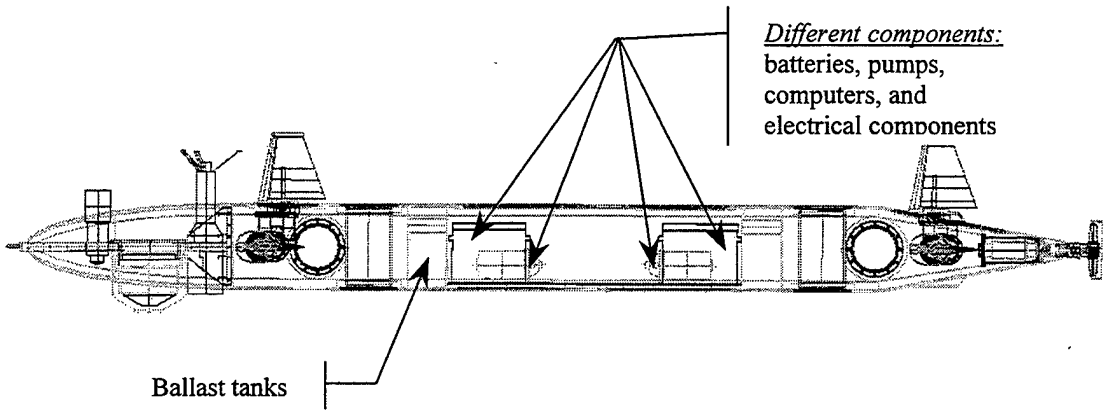

Figure II.4: New NPS AUV side view [Garibal, 1999].

Furthermore, two Pentium-based computer boards are going to be used to provide computational power. They are faster and use less power than the GESPAC combined to a Pentium chip used on the *Phœnix* AUV.

D. VIRTUAL WORLD

The main asset of an AUV, that is the ability of performing tasks autonomously, also becomes its worst drawback when it comes to in-water testing of the robot, because it operates in a remote and hazardous environment and it is often impossible to observe or communicate with the vehicle. The development process becomes consequently very difficult for the designers.

To overcome this problem, a virtual world was created which models salient characteristics of the ocean environment from the robot's perspective [Brutzman 94]. This allows developers to visualize robot behavior under diverse conditions.

Prior to the virtual world creation, the AUV had to be tested in the CAUVR laboratory test tank in order to verify any upgrade to the software. A simulation capable of modeling the AUV actions and reactions was therefore required and would make the project advance at a much faster pace.

The virtual environment provides an area of underwater terrain where the AUV can maneuver. The implementation of the *Phoenix* virtual world is divided into three major modules:

- vehicle hydrodynamics model,
- AUV software,
- interactive 3D graphics user window into the environment.

E. PREVIOUS WORK ON A GRAPHIC USER INTERFACE

Three years ago, a thesis will be written on the creation of a mission-generation expert system [Duane Davis, 1996]. The strategic level was a Prolog code and contained the mission of the AUV.

This expert system consisted of three distinct subsystems and a graphical user interface (GUI). The first subsystem was used to automatically generate missions by specification of overall mission goals. The second subsystem was a mission-specification facility that could generate arbitrarily complex missions phase-by-phase. The last

subsystem was an automatic strategic level code generator that creates Prolog or C++ programs using results from either of the other two subsystems. The GUI, automatic mission-generation facility, and mission-specification facility had been created on UNIX workstation.

F. SUMMARY

The Phoenix AUV is a high technology Autonomous Underwater Vehicle that can operate in shallow water. Using a tri-level software architecture RBM model, it mimics a manned submarine operational structure. The new AUV, which will be finished in September, should permit to increase the time autonomy and the performance by using a faster system. In order to accelerate its development, a virtual world has been created to perform virtual tests in various and unusual conditions.

Also, a mission-generation expert system was already created three years ago. It was able to generate the Prolog code used to define a mission in the Strategic level.

III. PROBLEM STATEMENT

A. INTRODUCTION

As described previously, the Phoenix is an Autonomous Underwater Vehicle (AUV). It means that it is independent to manage power and controls, is free to maneuver easily over large distance and depths, with little or no direct human supervision. However, it is understandable that it is necessary to define what is going to be the purpose of a mission. For this reason, it is useful to plan and define each mission before each experiment.

With the evolution of the technology, a script file appears on the way to define the mission planning. This file is in fact a text file with its own syntax and its own rules, which are often restricting. In order to solve the problem of writing this text, a Graphic User Interface (GUI) is a good solution.

B. EVOLUTION OF HARDWARE AND SOFTWARE ON THE NPS AUV

Three years ago, the code of the Phoenix AUV used two computers: a Gespac 68030 computer and a Sun Voyager Work Station.

The strategic level defines the different steps of the mission and decides the next step if a failure appeared. Its code was generated in Prolog code and was executed on the Sun Voyager [Duane Davis, 96].

Now, the Phoenix has a QNX which has replaced the Sun Voyager Work Station. However, the QNX doesn't have a Prolog compiler. So, another way to communicate with the AUV runs on the Phoenix. It consists to send a script file to the execution level.

On the new AUV, two Pentium processors are going to be used in order to obtain a faster system. Also, the code is going to be rewritten and the Prolog language won't be used anymore in the strategic level. Nevertheless, the use of the script file will continue to be the way to send the description of the mission to the robot.

C. HOW THE SCRIPT FILE WORKS

The script file is in fact a simple text file. This file is created on a laptop computer. Then, before a mission, the use of a point-to-point protocol via a radio communication permits to obtain a link between the laptop computer and the QNX, inside the robot. At this point, the script file is transmitted directly to the QNX, which can start the mission.

As said previously, the script file is a simple text file. In this file, each line represents a step of the mission. So, when the AUV receives the script file, the code starts reading it by the top and executes the order. If the first step succeeds, it is going to read the next line. The computer is going to continue until the last line, which constitutes the end of the mission.

In fact, there are two types of lines:

- The lines that contains a "#" as first character are simple lines of comment and are not interpreted by the code.
- The other lines are command lines and are used in the file ***exec.c***. These lines contain first a keyword, which permits to the code to know what kind of order is sent. And some of these keywords have to be associated with some values. These values have to be written after the keyword. In fact, they will be the parameters followed by the robot (Position, speed...).

Appendix A presents examples of script files, with an explanation of all the keywords used in order to create the script.

A script file contains generally two parts. The first part is the initialization and stops usually with the keyword "INITIALIZATION_DONE." This section contains some steps that have to be done before moving the robot.

The second part of the script defines how the robot is going to move and to control its position.

There are two types of control and, so, two keywords associated:

- "USE_WAYPOINT_CONTROL" defines a control of the robot with its position. Some coordinates define the points that the robot has to join.
- "USE_TIME_BASED_CONTROL" defines a control that uses the time as a reference. Here, the robot has to keep a heading and a depth during a period of time.

These two kinds of control are completely different and cannot use together in the same script file.

D. INTEREST OF A GRAPHIC USER INTERFACE (GUI)

The text file that is transmitted to the robot is very simple. But, it is also very simple to make a typing error inside and to have some problems. In fact, a simple mistake in the spelling of a keyword is going to be unreadable for the C program, inside the robot.

Also, it is always the same file that is sent to the robot. So, when it is decided that the mission planning is going to change, it is easy just to put a # sign before the keywords that have to be unable. But, the fact to forget to put this sign at the beginning of the line is going to transform a simple comment into a command.

Consequently, with these kinds of mistake, the script can contain some orders that are not usable together.

A similar problem can happen with embedded numeric values. Indeed, no boundary can be defined with the script file. So, the values sent to the boat may be impossible to execute.

Others problems of the script file is that it is not easy to follow the chain of the commands and it obliges to imagine what is going to be the way of the robot during the mission. This can be an easy exercise for a simple mission, but, for a long and complex mission, it becomes complicated.

Thus, the creation of a graphic User Interface that generates automatically the script file seems to be able to answer to all those problems. Indeed, the purposes of the GUI is:

- To avoid the user to type all the keywords and, so, not to make mistake in the spelling.
- To permit to define limits for the values and to avoid to enter values physically impossible.
- To create a way to evaluate the trajectory of the robot during the mission
- To assure that the commands are not contradictory or that the order of the commands is logical.

For this project, it was decided to program this Graphic User Interface with the Prolog Language and with the help of a Freeware version of Visual Prolog.

E. SUMMARY

This chapter shows how the evolution of the technology obliged to change the way to control the mission of the AUV. Now, the planning mission is defined with a text file, which is sent to the robot before the beginning of the mission. But, despite its simplicity, it is very easy to make mistakes because it is just a typing text. Then, the use of a Graphic User Graphic seem to be able to solve all of the problems generated by a manual action and, so, to build this file faster and more safely.

IV. PROLOG LANGUAGE

A. INTRODUCTION

The Prolog language was created in order to provide a logical programming language easy to understand and to learn. Contrary to the classic procedural languages, Prolog uses a logic deduction to find a solution to a given problem. In fact, it combines the different logical predicates known to be true to obtain all the possible solutions. Prolog provides also a easy and understandable syntax.

B. BACKGROUND

Prolog is the result of many years of research work. The first official version of Prolog was developed at the University of Marseilles, France, by Alain Colmerauer in the early 1970s as a tool for PROgramming in LOGic. The result was a language far more powerful than even today's well known programming languages, like Pascal and C. A Prolog program for a complex application will typically require only one tenth as many program lines as the corresponding C++ program.

Today, Prolog is a very important tool in programming artificial intelligence applications and in the development of expert systems. The demand for more "user friendly" and intelligent programs is another reason for Prolog's growing popularity. But the most important benefits of Prolog apply equally well to any application domain: By allowing the programmer to model the logical relationships among objects and processes, complex problems are inherently easier to solve, and the resulting program is easier to maintain through its lifecycle.

C. DIFFERENCE WITH OTHER LANGUAGES

Prolog is what is known as a declarative language. This means that given the declaration of necessary facts and rules, Prolog uses deductive reasoning to solve programming problems. This is in contrast to traditional computer languages (such as C, BASIC and Pascal) which are procedural languages. In a procedural language, the programmer must provide step-by-step instructions that tell the computer exactly how to solve a given problem. In other words, the procedural programmer must know how to solve the problem before the computer can do it. The Prolog programmer, on the other hand, only needs to supply a description of the problem and the ground rules for solving it. From there, the Prolog system is left to automatically determine a solution.

Because of this declarative (rather than procedural) approach, well-known sources of errors such as loops that carry out one too many (or one too few) operations are eliminated right from the start. Prolog encourages the programmer to start with a well-structured description of the problem; so that, with practice, Prolog can also be used as both a specification tool, and the implementation vehicle for the specified product.

D. PROGRAMMING IN LOGIC

In Prolog, the system arrives at solutions by logically inferring one thing from something already known. Typically, a Prolog program isn't a sequence of actions: it's a collection of facts together with rules for drawing conclusions from those facts. Prolog is therefore a declarative language.

Prolog includes an inference engine, which is a process for reasoning logically about information. The inference engine includes a pattern matcher, which retrieves stored (known) information by matching answers to questions. Prolog tries to infer that a hypothesis is true (in other words, answer a question) by questioning the set of information already known to be true. Prolog's known world is the finite set of facts (and rules) that are given in the program.

One important feature of Prolog is that, in addition to logically finding answers to the questions you pose, it can deal with alternatives and find all possible solutions rather than only one. Instead of just proceeding from the beginning of the program to the end,

Prolog can actually back up and look for more than one way of solving each part of the problem. This technique is called backtracking.

Predicate logic was developed to easily convey logic-based ideas into a written form. Prolog takes advantage of this syntax to develop a programming language based on logic. In predicate logic, you first eliminate all unnecessary words from your sentences. You then transform the sentence, placing the relationship first and grouping the objects after the relationship. The objects then become arguments that the relationship acts upon. In others words, a sentence of the form "Subject relationship Object" becomes a predicate logic clause of the form " relationship(subject, object)."

For example, the following sentences are transformed into predicate logic syntax:

Natural Language:	**Predicate Logic:**
A car is a vehicle. A rose is red.	`vehicle(car).` `red(rose).`
Bill likes a car if the car is fun.	`likes(bill, Car) if fun(Car).`

E. USING PROLOG

1. Prolog Syntax

A Prolog program is made up of *clauses*, which conceptually are two types of phrases: facts and rules.

- *Facts* are relations or properties that the programmer knows to be true.
- *Rules* are dependent relations; they allow Prolog to infer one piece of information from another. A rule becomes true if a given set of conditions is proven to be true. Each rule depends upon proving its conditions to be true.

Prolog facts have the general form:

```
relation(object1, object2, ..., objectN)
```

Rules have two parts: a head and a body separated by the special :- token.

- The *head* is the fact that would be true if some number of conditions were true. This is also known as the conclusion or the dependent relation.

- The *body* is the set of conditions that must be true so that Prolog can prove that the head of the rule is true.

So, rules have the general form `Head:- Body.`

```
relation(object,object,...,object):-
  relation(object,...,object),
    .
    .
  relation(object,...,object).
```

In fact, each *relation* is called a predicate in Prolog.

The programmer is free to choose names for the predicates in his programs. Syntactically, predicates must begin with a lower-case letter, followed by any number of characters; characters are upper-case or lower-case letters, digits, and underscores.

Variables permit the programmer to write general facts and rules, and also to ask general questions.

Variable names in Visual Prolog must begin with a capital letter or an underscore character (_), after which you can use any number of letters (upper-case or lower-case), digits, or underscores.

Variables in Prolog get their values by being matched to constants in facts or rules. Until it gets a value, a variable is said to be "free;" when it gets a value, it becomes "bound."

2. Structure of Visual Prolog Programs

A Visual Prolog program has the following basic structure:

```
DOMAINS
  /* ...
  domain declarations
  ... */
PREDICATES
  /* ...
  predicate declarations
  ... */
CLAUSES
  /* ...
  clauses (rules and facts)
  ... */
GOAL
  /* ...
   subgoal_1,
   subgoal_2,
   etc. */
```

The **clauses** section contains the facts and rules that Visual Prolog will operate on when trying to satisfy the program's goal.

The **predicates** section contains the declaration of the predicates (*relations*) and the domains (types) of the arguments to these predicates. Predicate declarations are of the form

```
PREDICATES

predicateName(argument_type1, argument_type2, ..., argument_typeN)
```

argument_type1, ..., *argument_typeN* are either standard domains or domains that have been declared in the **domains** section. Declaring the domain of an argument and defining the argument's type are the same thing.

The **domains** section contains the declaration of any nonstandard domains that are being used for the arguments to the predicates. Domains in Prolog are like types in other languages. Visual Prolog's basic standard domains are ***char***, ***byte***, ***short***, ***ushort***, ***word***, ***integer***, ***unsigned***, ***long***, ***ulong***, ***dword***, ***real***, ***string***, and ***symbol***.

The **goal** section contains the program's *goal*. It is simply a list of subgoals. In order to terminate the progam, all the subgoals in the goal section have to be satisfied. Subgoals and goals are satisfied by binding various combinations of variables while satisfying predicate relations.

3. Unification and Backtracking

Prolog facts and rules receive information by being called with arguments that are constants or bound variables; they return information to the calling procedure by binding variable arguments that were unbound.

Unification is the process used by the Prolog inference engine for matching two predicates and assigning free variables to make the predicates identical. This mechanism is necessary so Prolog can identify which clauses to call and bind values to variables. These are the major points about matching (unification) presented in this chapter:

- When Prolog begins an attempt to satisfy a goal, it starts at the top of the program in search of a match.

- When a new call is made, a search for a match to that call also begins at the top of the program.
- When a call has found a successful match, the call is said to *return*, and the next subgoal in turn can be tried.

Once a variable has been bound in a clause, the only way to free that binding is through backtracking.

Backtracking is the algorithmic mechanism that instructs Prolog where to go to look for solutions to the program. This process gives Prolog the ability to search through all known facts and rules for a solution. These are the four basic principles of backtracking:

- Subgoals must be satisfied in order, from top to bottom.
- Predicate clauses are tested in the order they appear in the program, from top to bottom.
- When a subgoal matches the head of a rule, the body of that rule must be satisfied next. The body of the rule then constitutes a new set of subgoals to be satisfied.
- A goal has been satisfied when a matching fact is found for each of the extremities (leaves) of the goal tree.

A call that can produce multiple solutions is *non-deterministic*, while a call that can produce one and only one solution is *deterministic*.

Visual Prolog provides three tools for controlling the course of a program's logical search for solutions: these are the two predicates ***fail*** and ***not***, and the ***cut***.

- The ***fail*** predicate always fails; it forces backtracking in order to find alternate solutions.
- The ***not*** predicate succeeds when its associated subgoal can't be proven true.
- The ***cut***, represented by a ! token in a program, prevents backtracking. This is useful if one and only one value for a predicate subclause is desired.

F. SUMMARY

This chapter provides a general presentation of Prolog. It permits readers to have a idea of the Prolog logic. Indeed, Prolog differs from traditional programming languages: it's a declarative language. Instead of series of steps specifying how the computer must work to solve the problem, a Prolog program consists of a description of the problem.

Prolog has a very short and simple syntax, which is much easier to learn than those of more traditional programming languages. In fact, Prolog was a tool particularly adapted to the creation of our Graphic User Interface.

V. VISUAL PROLOG ENVIRONMENT

A. INTRODUCTION

Visual Prolog Personal Version is a freeware version of Visual Prolog Version 5. This version doesn't permit to distribute any application created with. However, it permits to learn how to program a GUI and to use it personally.

Visual Prolog provides several tools that permit the user to create an application easily. It has an interactive Visual Development Environment (VDE) which includes text and various graphical editors, code generating tools (Experts), build control logic, and extensions to Prolog in the form of a Visual Programming Interface (VPI). It also includes a Prolog compiler, various include files and libraries, a linker, and various example and help files.

B. PRESENTATION OF THE VISUAL DEVELOPMENT ENVIRONMENT

The Visual Development Environment (VDE) combines the compiler with an editor, a resource toolkit, a resource and application Expert, an interactive make facility and various browsing facilities.

After the interactive visual creation of the user interface components, a running prototype is automatically generated. The application Expert creates all the necessary files for a project, and the resource Expert knows how to generate the Prolog code to support all the selected resources.

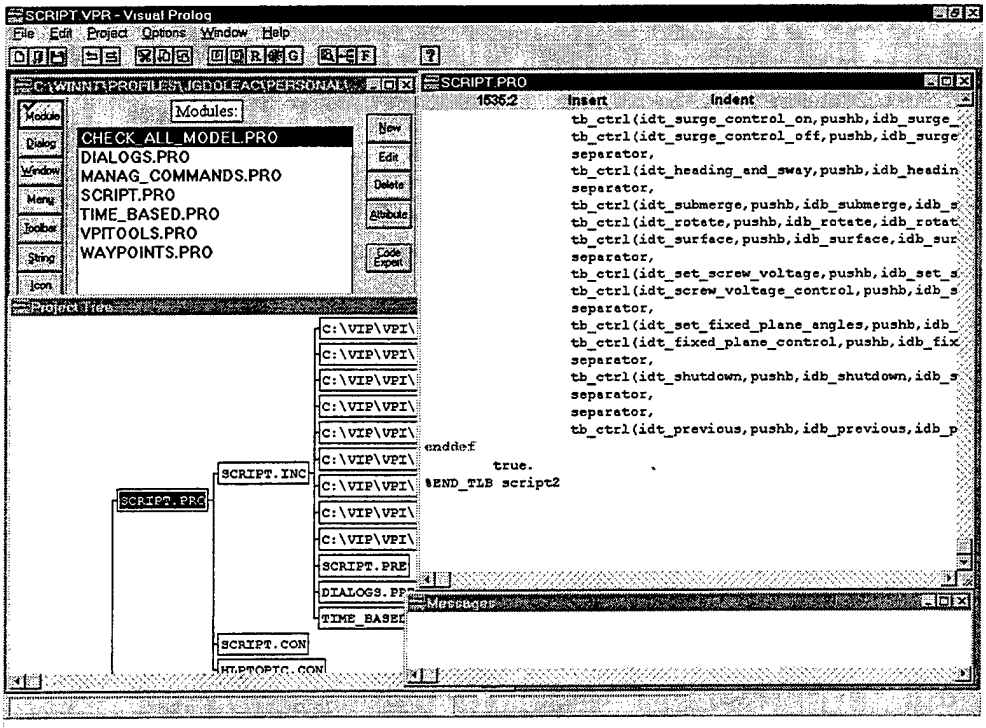

Figure V.1: The Visual Development Environment for Visual Prolog.

C. THE APPLICATION EXPERT

The Application Expert is a tool to help in the creation of a new Visual Prolog Project, as well as to help to change certain settings in this Project later.

A GUI application requires a lot of footwork just to get started. It is useful to create a large chunk of startup code, many standard event handlers, usually an About Box, as well as resources, menus, etc. These standard tasks might easily take a couple of days to do from scratch. The Application Expert helps to perform all these operations, based on some options it is necessary to set.

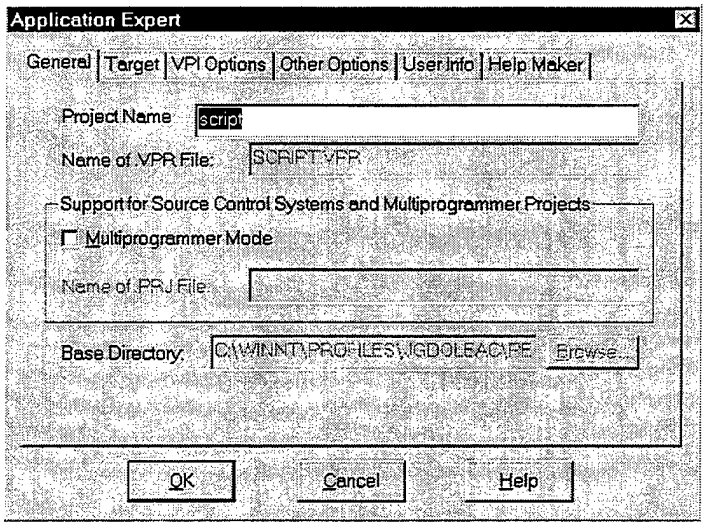

Figure V.2: The application Expert dialog box.

D. THE PROJECT WINDOW

When a project is opened, this window will automatically appear. If the Project window is closed, the project will be closed. The Project window contains lists of all the components of a Visual Prolog application by type of component.

Figure V.3 shows the Project window. Clicking on a selection at the left side of the window changes the content of the list box in the middle of the window. This list box shows all components of the project that correspond to the component type of the active left button.

One component can be selected from the list, and a click on a button on the right side activates the tool you need to work on the current component. A double-click on the component name activates the editor for that type of component.

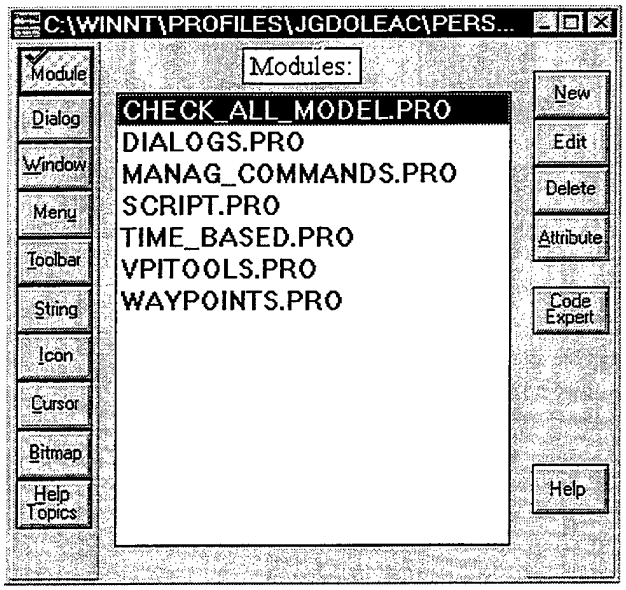

Figure V.3: The Project Window.

1. Module

The list box contains a list of the source modules in the current project. When activating a **Project | Build** these files will be compiled. When you double click on a source file in the list, the text editor will open that file. The source modules are stored in ordinary files in the underlying file system.

2. Dialog and Window

These list boxes contain lists of the dialogs and windows in the project. Double clicking on a dialog or a window brings up the Editor set to edit that dialog or window. In fact, the Window Editor and the Dialog Editor are very closely related.

The Window and Dialog Editor permit to create whatever form of wished dialog. The layouts of all dialogs are stored entirely in the project .VPR file. The main purpose of the Dialog Editor is to define and layout the controls inside a dialog.

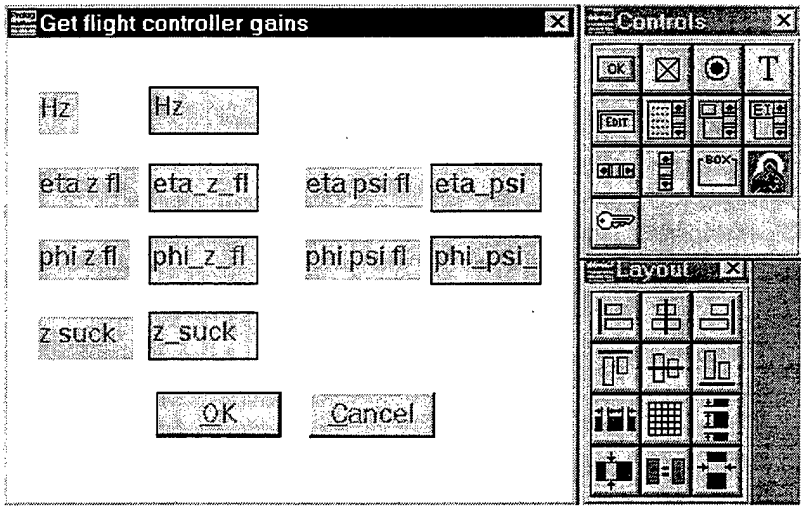

Figure V.4: The Dialog Editor.

3. Menu

The list box contains a list of the menus in the project. These can be used as either pull-down menus associated with windows, or as pop-up menus. Double-clicking brings up the menu editor.

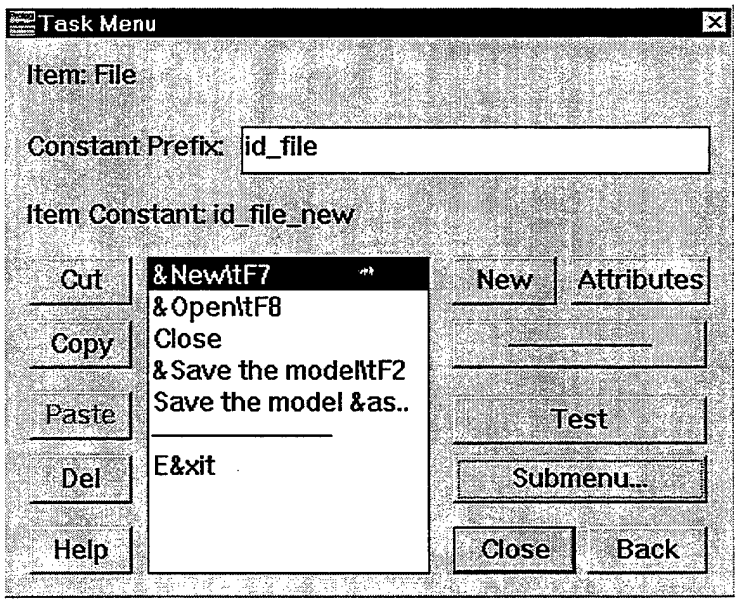

Figure V.5: Editing a menu.

The Menu Editor can be used to create menus that can be used as both pull-down menus for windows, and as pop-up menus for object oriented user interfaces. When a

menu has been created in the Project window, double clicking on it will bring up the Menu Editor.

4. Toolbar

Clicking on **new** to create a new toolbar allows to specify what style the toolbar should have (left, bottom, right, inside, moveable), and what the background color of the toolbar should be.

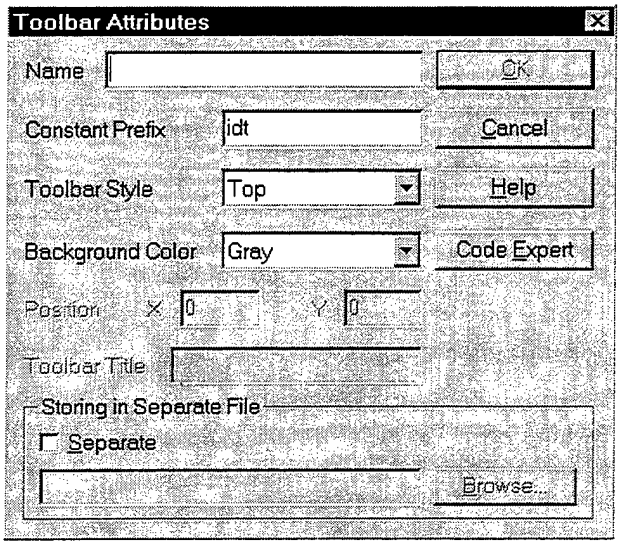

Figure V.6: The Toolbar Attributes dialog.

Clicking on one of the previously registered toolbars brings up the Toolbar Editor. It shows the toolbar as the application will show it. There are several kinds of control that can be added into the toolbar: Push Button, Check Button, List Button, Static Text, Context Sensitive Text and Separator.

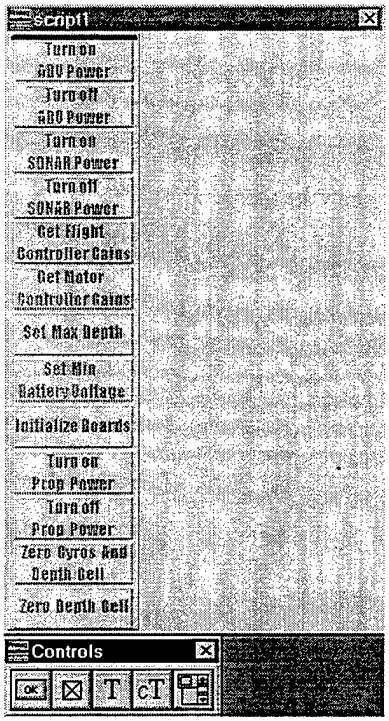

Figure V.7: The Toolbar Editor (with Push Buttons).

For each of the controls, another window permits to specify the properties of the control. For example, in order to create a Push Button, the user has to inform the system on what Bitmap images are going to represent the button.

5. Bitmap, Icon and Cursor

The Graphics Editor is a convenient tool for creating, viewing and editing icons, cursors and small bitmaps. The images can be passed to the Windows Clipboard or saved in files.

The Graphics Editor allows to create and edit images ranging in size from 4x4 pixels to 64x64 pixels, by using either a 16-color palette or monochrome shadow palette.

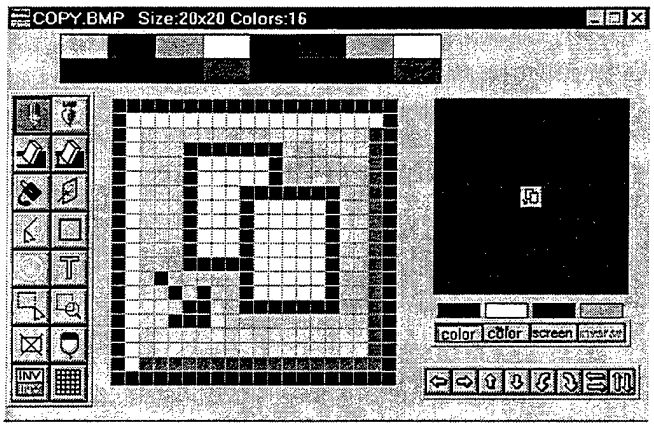

Figure V.8: The Graphic Editor.

The Graphics Editor is invoked from the Project window for Icons, Cursors and Bitmaps. However, when you register large bitmaps in your project, Visual Prolog will call the Windows Paintbrush program instead to edit the bitmap.

E. THE CODE EXPERTS

There are several Code Experts in Visual Prolog. We have already seen the **Application Expert**, which generates all the default code, resources and make-scripts for an application.

The other Code Experts are:

1) the **Dialog and Window Expert**,
2) the **Dialog Package Expert** and
3) the **Toolbar Expert**.

These three code experts are used for generating Prolog source code after the layout of resource components has been created.

The advantages of the Code Experts are that:

1) it save a lot of typing,
2) it gives a standardized way of handling things,
3) by using the code experts, the user can easily come to the source code through selection of the user interface component.

A slight disadvantage is that the code experts insert some extra comments, which they use to be able to locate the generated source code at a later stage.

The Code Experts can also automatically update the source code that they generate when the layout or attributes of a user interface component are modified.

1. Dialog and Window Expert

The Dialog and Window Expert is the tool which connects Prolog code to the layout of windows and dialogs. After a dialog or a window is designed, the Dialog and Window Expert can be used to insert the necessary Prolog code to manage window and dialog creation and event handling.

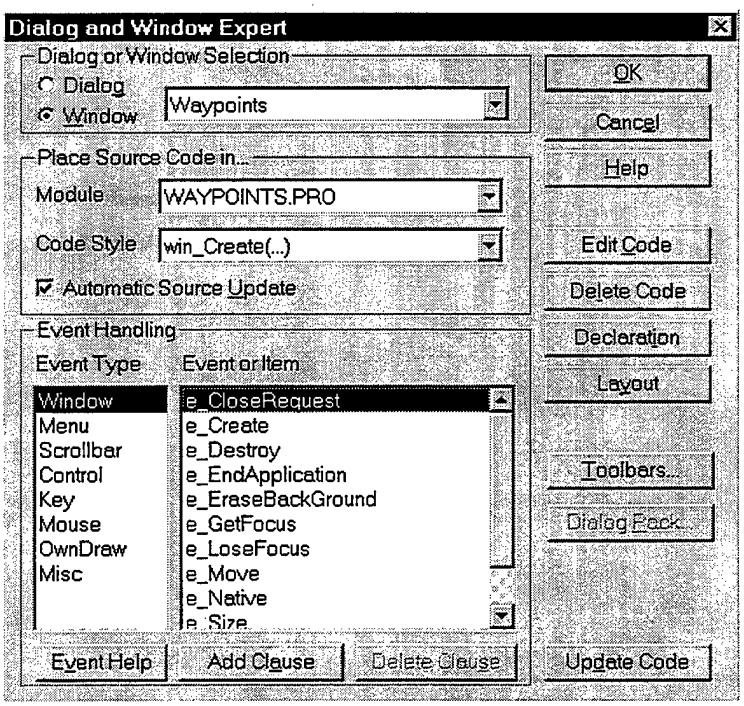

Figure V.9: The Dialog and Window Expert.

2. The Dialog Package Expert

The Dialog Package makes it easy to initialize and retrieve the values for a dialog, and it has a number of features for handling and validating the control values. The Dialog Package Expert makes it possible to specify the options for the Dialog Package in some dialogs.

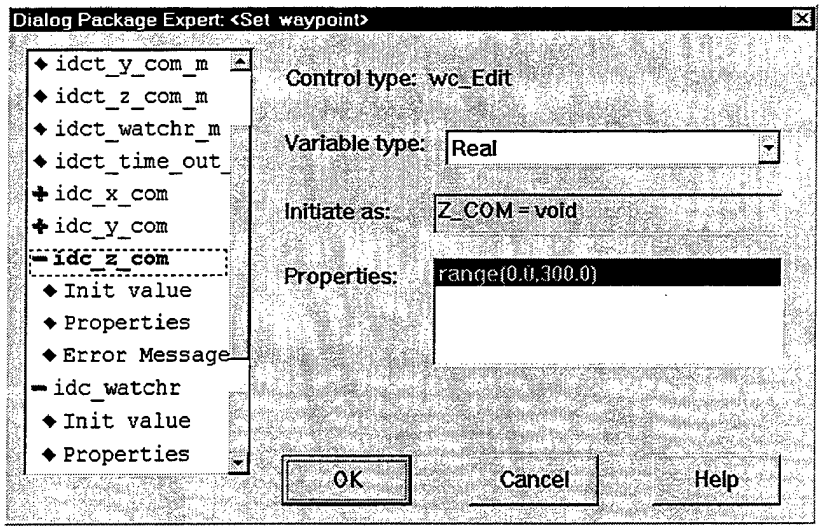

Figure V.10: The Dialog Package Expert.

3. The Toolbar Expert

For each toolbar created from the Project window, a Prolog predicate is needed to actually create the toolbar. The source for this predicate can be generated by the Toolbar Expert.

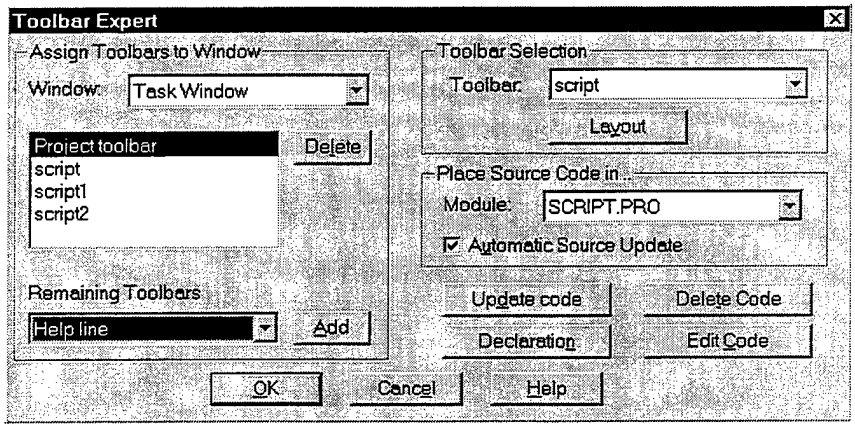

Figure V.11: The Toolbar Expert.

F. SUMMARY

This chapter presents different tools that are available on Visual Prolog. This software is very useful to create a Graphic User Interface. Indeed, it can generate automatically all the basic code for the creation of a new project and for the dialogs and windows associated. A lot of other facilities are also provided to help the user to understand how the code is running and to debug errors.

VI. AUV SCRIPT USER INTERFACE CODE

A. INTRODUCTION

The first design goal of the Graphic User Interface (GUI) is to provide a tool that represents graphically the chain between the keywords, and thus avoids the user typing these keywords. Then, different problems appeared in the creation of the Prolog code.

The purpose of this chapter is to explain the important steps that were followed during this project. The complete Prolog code is provided in *Appendix B*.

B. STORING INFORMATION

The first step of the creation of the Graphic User Interface is to find a way to store the keywords that the user wants to insert in the script file. So, the use of a database was obvious. The purpose of this database is also to make easier the drawing of the model.

This database is declared in the file **Script.inc** as below.

```
global domains
     messages = message(text, values)
     text = string
     values = DIALOG_REAL*
     posit = integer

global database - keywords
     keyword(posit,messages,COLOR)
     determ counter(Integer)
```

The database is declared as a global database, which permit to use it in all the programs of the project.

In fact, it contains two elements:

- `counter(Integer)` is a counter that permits Visual Prolog to store the number of keywords created. It is incremented when a new keyword appeared.
- `keyword(posit,messages,COLOR)` is used to store all the keywords. Each time a keyword is created, the database stores:
    1. its position under the variable posit,
    2. the text and the values under the variable messages,
    3. the color that is used to draw the keyword in the graphic interface.

There is in fact one keyword that is gray, the other one are white. This is used to locate the active keyword. If the user wants to insert a new keyword, it will go to appear after the "gray" keyword.

C. VISUALIZING THE INFORMATION

Visual Prolog provides a lot of predicates that are useful for managing the graphic window. One of the most important functions of the program, which is in the file **script.pro**, was created with these predicates. Indeed, this function permits to draw inside the GUI. It is called with the predicate `winRefresh()` each time the widow need to be redraw :

- Insertion of a new keyword,
- Deleting a keyword,
- Resizing of the window,
- Moving the scrollbar.

First, this predicate clears the window, then it calls another predicate `dessin()`. This one is in fact a loop that is going to use the database and, for each position; it going to draw a rectangle with the keyword written inside.

D. USING THE BUTTONS

As we saw previously, Visual Prolog provides a tool that permits to create easily a toolbar with all the buttons. Also, the application expert generates automatically the predicate called with a click on a button. At this point, it becomes necessary to define what is going to be the next step.

In our case, each button defines a keyword. The predicate `build_keyword()` (**Script.pro**) is called by each buttons. This predicate is going to locate the keyword associated with the color Gray in the database. Then it is going to insert the new keyword in the database and give it the position after this "gray" keyword. Finally, a call to the predicate `winRefresh()` is going to redraw the window, with the new keyword.

E. BUILDING A TEXT FILE

The main purpose of the GUI was the generation of a text file. The problem was there to interpret what was stored inside the database and to write it in a text file.

The predicate `openwrite()` permits to open a text file for writing inside. Then with the use of the write function, the desired text is printed. The same kind of loop as in the drawing function has been used here. It consists to take the keywords one by one in the database, to pick up the text and the values, to modify the text to be conform to the script file syntax, and to write it in the text file followed by the associated values.

F. SUMMARY

Even though Visual Prolog has tools that are very useful to create easily and quickly a lot of code, it doesn't make all the code. This chapter describes the structure of the Prolog code and how it is constructed. However, only the main points that were essential for the creation of the GUI are explained here. Full details are provided by the source code in Appendix B.

VII. USING THE AUV SCRIPT GRAPHIC USER INTERFACE

A. INTRODUCTION

The main purpose of this Graphic User Interface (GUI) is the generation of the script file, which is a simple text file (**mission.script**). So, it permits the user to easily and quickly create this file without errors of syntax. This chapter explains the way to start the GUI, to use its different menus, and to produce a mission script.

B. STARTING WITH THE GUI

Script

The GUI starts with a simple double click on the icon "Script".

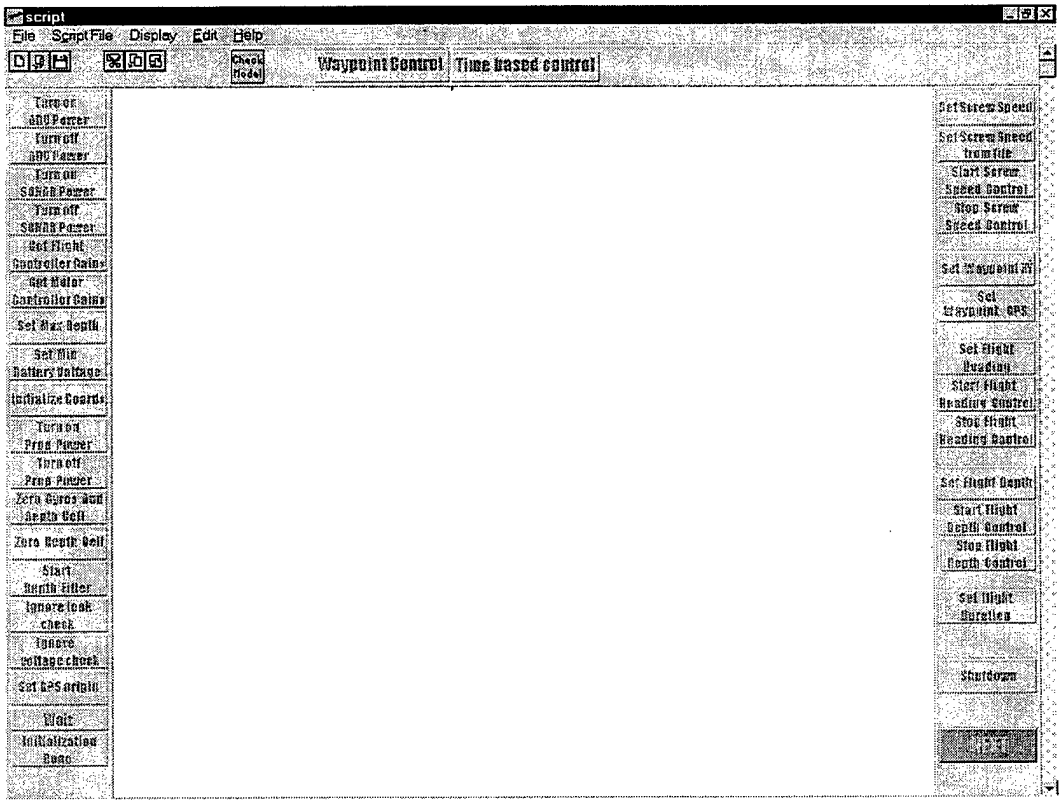

Figure VII.1: Starting the GUI.

The first window is as the figure above, which shows the environment of the GUI. At this point, the buttons are not enabled. The user has just the choice between creating a new file, opening an existing file (***.bin**) or opening a script file (***.script**).

There are also two menus available which permit access to the same option:
The task menu is available at the top of the screen.

Figure VII.2: The main menu.

The pop-up menu is enabled with a click on the right button of the mouse, and provides the functionality of the Edit, Display, Script File and File pull-down menus, respectively.

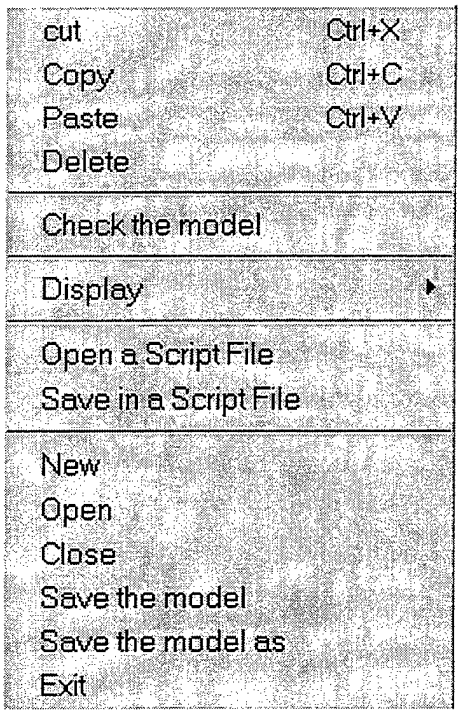

Figure VII.3: The pop-up menu.

C. USING THE MENU TO MANIPULATE THE FILES

1. Creating a New Model

When the user selects the menu **File ¦ New**, the initialization buttons on the left of the window are automatically enabled. Two buttons on the top of the window, "WAYPOINT CONTROL" and "TIME BASED CONTROL," are also available.

The user can now select the buttons desired and a rectangle with the corresponding keyword is automatically created on the screen. The number on the left side of each rectangle indicates its position.

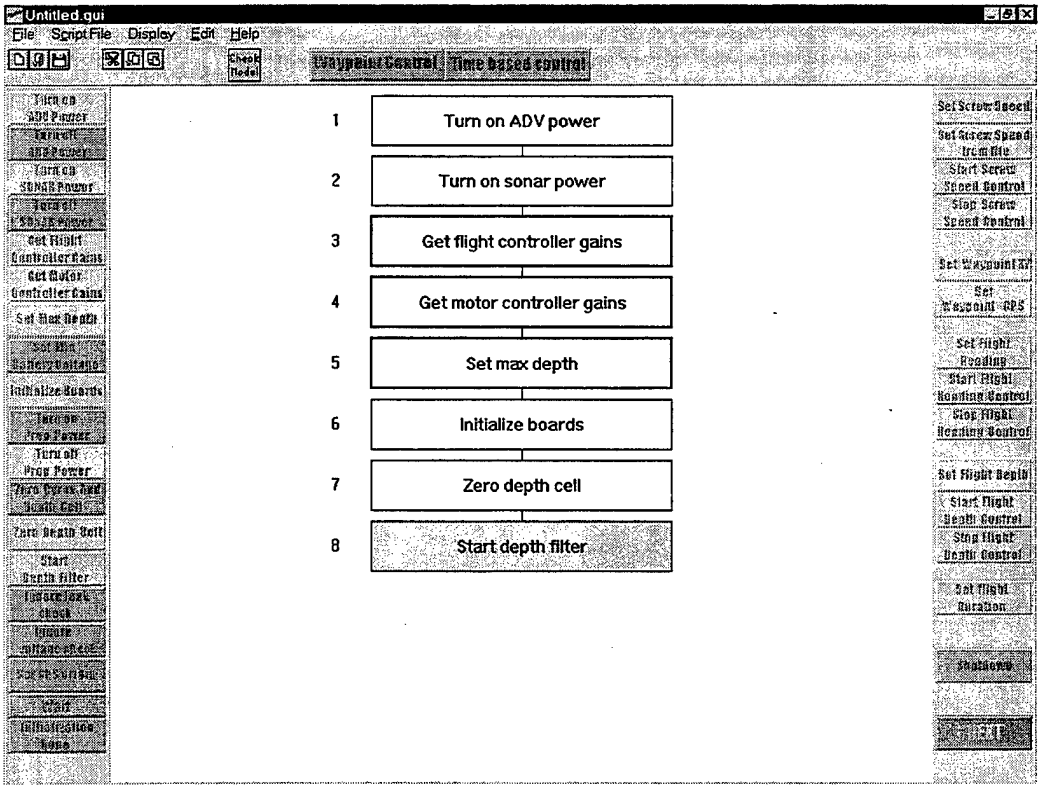

Figure VII.4: GUI Environment with example mission script commands.

2. Saving a Model

When the user has finished creating a model, he simply has to select **File | Save the model** in the task menu. The file is going to be saved with a "**.gui**" file extension.

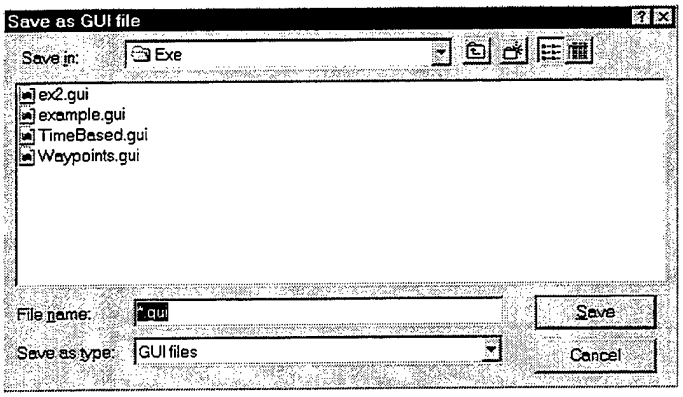

Figure VII.5: Saving a script in GUI format.

It is important to note that even though the model had been saved, this operation is just a way to keep the script for future use. So, it doesn't mean the script file has been created. For this, another option exists (See below in "Saving a script file").

The option **File |Save the model as** is also enabled in the task menu.

3. Opening a Model

The user can also open a model that has already been created and saved before. These files have a file extension "**.gui**". The script with all the rectangles is directly displayed in the window and the enable buttons are relative to the opened model.

4. Opening a Script File

The user can also directly open a script file (***.script**) under **Script | Open a Script File**. In fact, this file is a simple text file.

Usually, the name of the script file is **mission.script**. However, the GUI allows the possibility for other name of the Script file but the extension is always "**.script**".

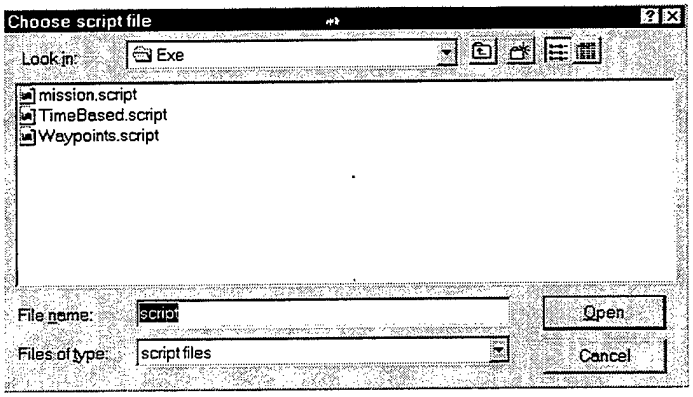

Figure VII.6: Opening a script file.

The principle of this operation is to read the text file and to store each line of the text in a database. The consequence is that, if there is a mistake in the text file, the same mistake is going to appear in the Graphic Environment.

5. Saving in a Script File

This operation is enabled under **Script | Save in a Script File.**

It takes the database and creates directly the text corresponding to each keyword and their associated values. If the file already exits, it is going to overwrite this text file.

The figure below shows what kind of text file can be generated with the use of the GUI.

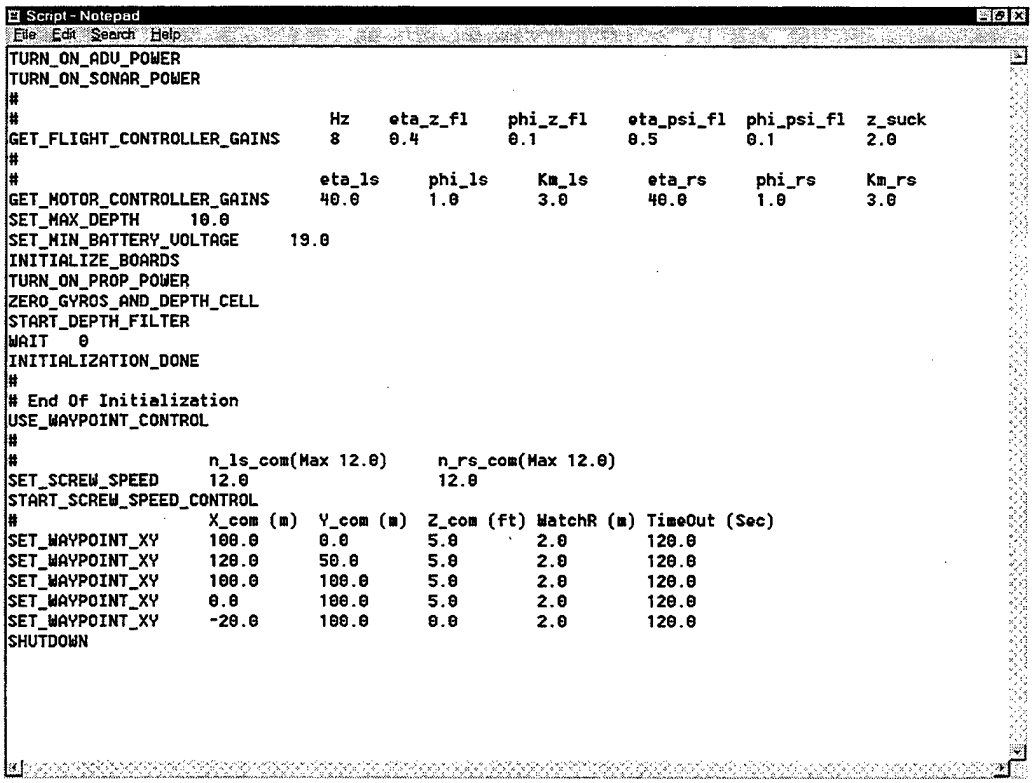

Figure VII.7: Example of script file generated by the GUI.

D. MANIPULATING THE MODEL

As shown previously, when the user creates or opens a model, or directly opens a script, some of the buttons become enabled. When the user clicks on a button, he creates a rectangle which contains the text associated with the button. But more manipulations are allowed.

1. Select a Rectangle

You can notice that there is always a rectangle that has a gray color which is the active rectangle. When the user clicks on a button, the corresponding rectangle is going to take place after this gray rectangle. As default, the active rectangle is the last rectangle of the chain. Of course, the user can decide where he wants to put the active rectangle by using the up and down buttons on the keyboard or by clicking with the left button of the mouse on the desired rectangle.

2. Enter the Values Associated With the Keywords

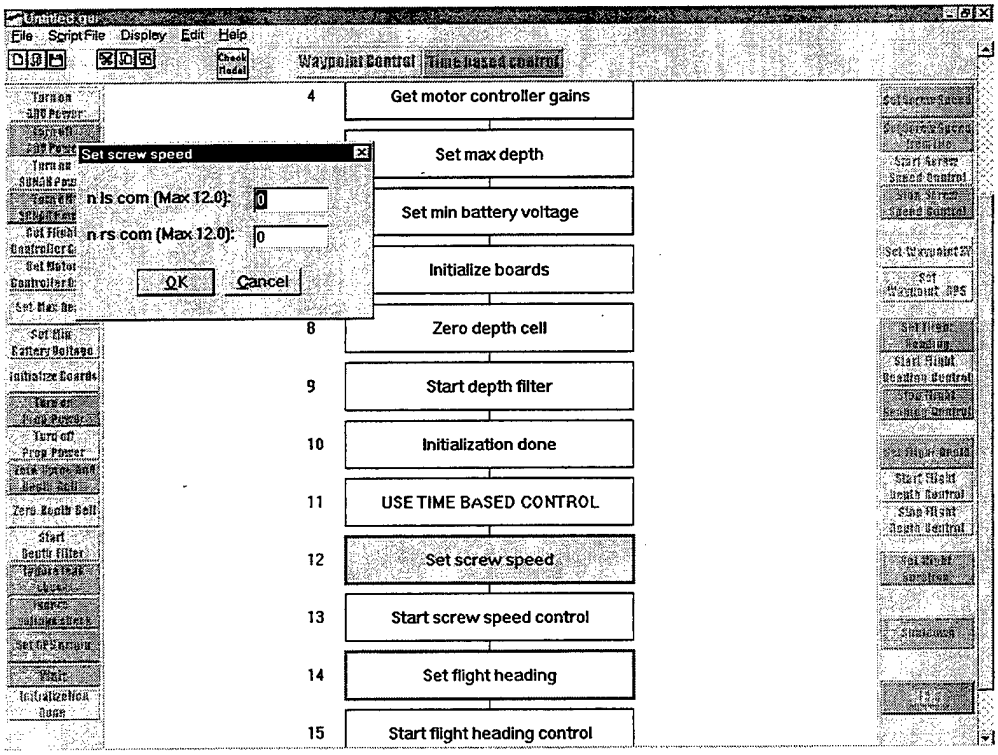

Figure VII.8: The dialog to enter the values.

Also notice that some of the rectangle have colored borders. This is to indicate that the keyword is associated with some values. There are two types of colored rectangle:

- ***red*** rectangles represent the keywords for which all the values are equal to zero.
- ***blue*** rectangles represent the keywords for which at least one value is different to zero. It generally means that the values have been modified at least one time.

By double clicking with the left button of the mouse on the rectangle or by selecting the rectangle and pressing Enter on the keyboard, the user can now enter the values, and then confirm OK. When the dialog box will be reopened, the values previously entered will be appeared.

3. Moving a Keyword

Usual editing functions, which can be found in most software, are also available here. Those functions are used to move and to reorganize the model.

The **Copy** and **Cut** keys, which are under **Edit | Copy** or **Cut**, in the pop-up menu or with the accelerators **Ctrl+c** and **Ctrl+x**, put a previously created keyword in the clipboard. Using the command **Paste** (under **Edit**, in the Pop-up menu or with the accelerator **Ctrl+v**), the user can now insert this keyword after the gray rectangle.

4. Use of "Waypoint Control" and "Time-Based Control"

The user has also the possibility to use the two buttons at the top of the window, "WAYPOINT CONTROL" and "TIME BASED CONTROL." However, when one of these buttons are selected the keywords associated appear with them on the screen. And, the other button is not enabled. This operation avoids having later, in the same script file, the keyword "USE_WAYPOINT_CONTROL" and "USE_TIME_BASED_CONTROL", which is inconsistent for vehicle control.

Notice that with the creation of one of those two keywords, the buttons associated with this type of control are enabled on the right side of the window. The buttons that are not enabled are either impossible to access with the kind of chosen control, or useless in the present state. For example, in order to have access to the button "Start screw speed

control," it is required to first define the keyword "Set screw speed" or "Set screw speed from file."

Also, the toolbar on the right side of the window has a blue button NEXT, which permit to change this toolbar and to have access to more buttons. On the new toolbar, a similar button named PREVIOUS permits to come back to the first toolbar.

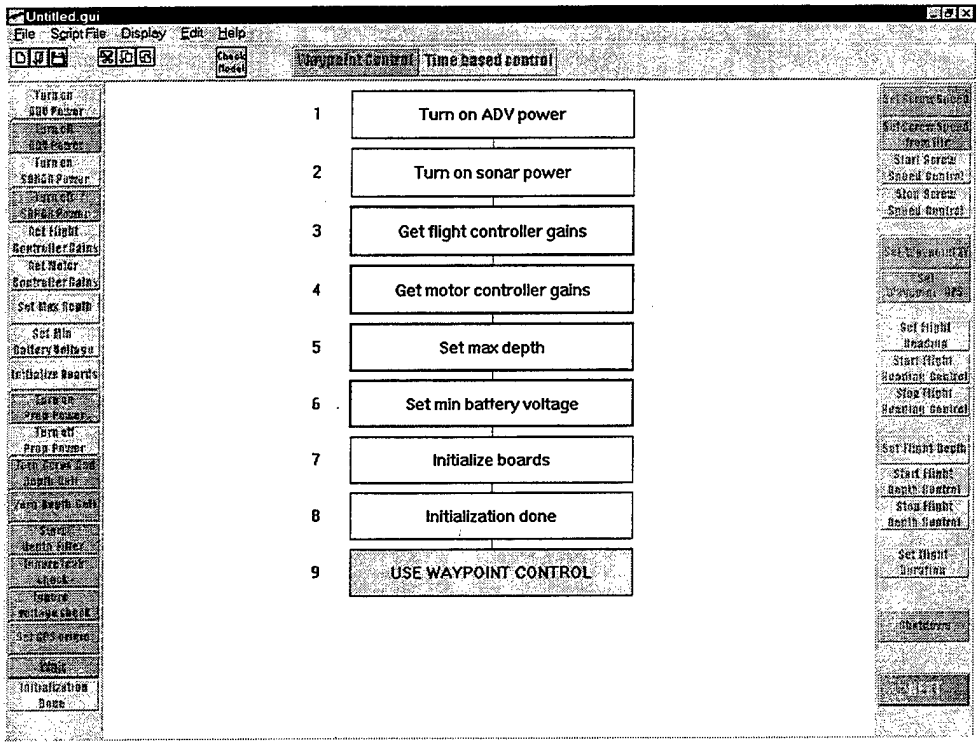

Figure VII.9: Selection between Waypoint control and Time-Based control.

a. Using GPS Control

When the user decide to use the waypoint control, he can then run the robot using the GPS system (longitude, latitude). The keyword "Set waypoint GPS" permits to enter in the script this kind of coordinates. In this case, the dialog associated permits the user to enter the latitude and longitude in degrees, minutes, seconds and milliseconds. This system is easier because the maps are in general written with these units. However, the Prolog code is going to convert those coordinates in milliseconds in order to write them in the script file.

b. Displaying Waypoint Control

If the user decides to use waypoint control, he can also use the keywords "Set waypoint XY." Each of these keywords represents a point where the robot has to go.

In the script file, it is represented as followed:

#	X_com (m)	Y_com (m)	Z_com (ft)	WatchR (m)	TimeOut (Sec)
SET_WAYPOINT_XY	100.0	0.0	5.0	2.0	120.0
SET_WAYPOINT_XY	120.0	50.0	5.0	2.0	120.0
SET_WAYPOINT_XY	100.0	200.0	5.0	2.0	120.0
SET_WAYPOINT_XY	0.0	100.0	5.0	2.0	120.0
SET_WAYPOINT_XY	-20.0	100.0	0.0	2.0	120.0

The robot is going to navigate through these points in the same order as they are written in the script file.

For this reason, under **Display | Waypoints**, a function exists which generates a window. In this window, the user can visualize an approximation of the commanded vehicle trajectory. It can help to detect an error when the values were entered.

At each point, the waypoint number and coordinates are displayed with a red circle around the point. This circle represents a sphere with the watch radius (WachR).

All the waypoints are going to be represented on this window. However, with a click on the left button of the mouse, the user can zoom on the area around the clicked point. With a double click on the left button of the mouse, the original figure will be redrawn.

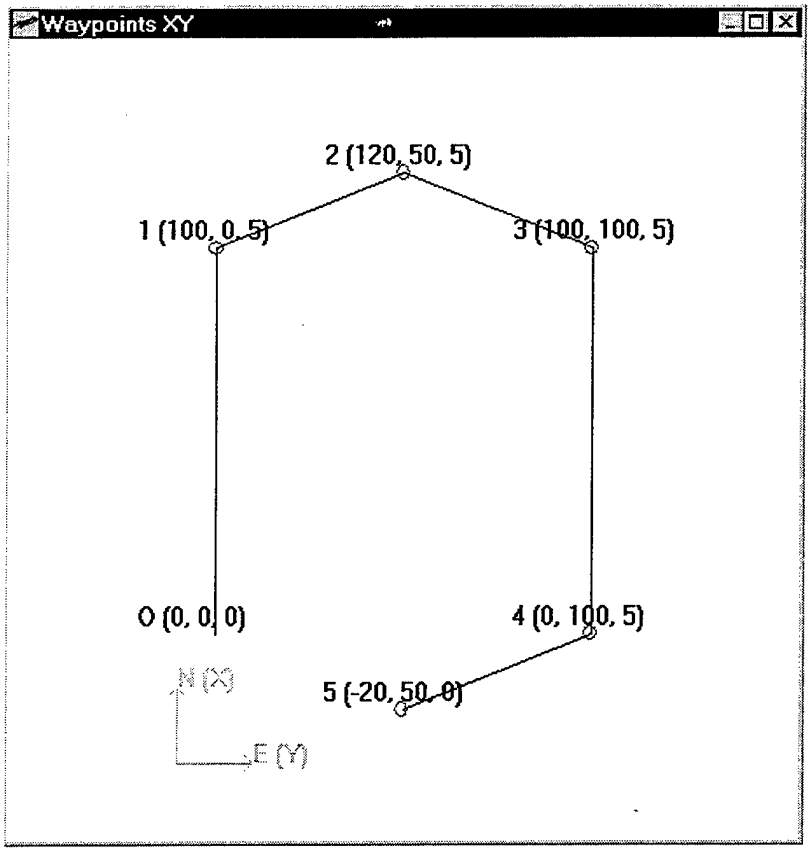

Figure VII.10: Waypoint control Display.

Note: For AUV physical stability reasons, there is a check function that prevents entering two waypoints less than 10 meters apart.

c. Displaying Time-Based Control

For time-based control, the robot is going to move when it receives an order of time ("Set flight duration"). At this point, it is going to use the last depth and heading entered in the script. Under **Display | Time Based Flights**, the user can open a window that is going to show an approximation of the commanded trajectory of the robot.

This display shows the trajectory of the robot step by step. Each step is written with the heading, the depth and the time. In this case, the red circle indicates the end of each flight. On this example, the third step shows a part where surge control is used for 20 seconds.

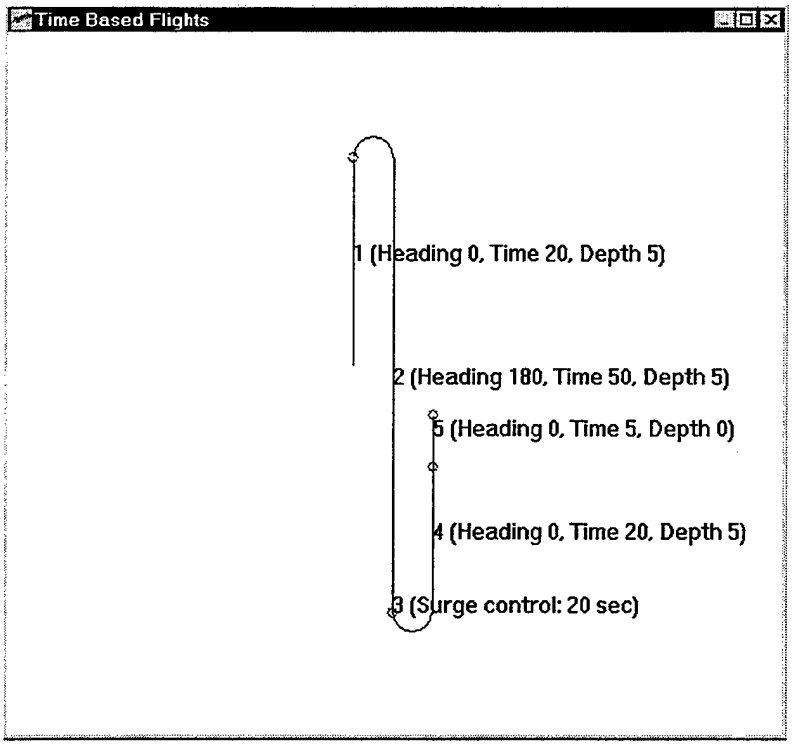

Figure VII.11: Time based control display.

E. CHECKING THE MODEL

To permit the user to build a right script file, some functions was added. First, there is a restriction in the use of the buttons. If the user wants to use some keywords, some conditions have to be right before. For example, if the user wants to use the button **"Start heading control,"** then the keyword **"Set flight heading,"** which defines an angle for the heading, has to be used at least one time.

Another application is the creation of a check function, which offers the possibility to look at each keyword in the model and to check if it is in a logical place. That permits the user to know if the script file is going to be right and understandable for the AUV.

If there is a mistake in the script, an error message will appear. This message tells the user that, at the place specified, it is not logical to insert the specified keyword. To help to find the error, the number in brackets indicates the position of the keyword in the model.

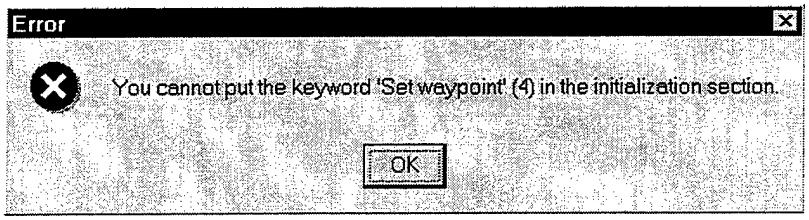

Figure VII.12: Sample error message.

F. SUMMARY

The Graphic User Interface allows creating a model. This model is displayed as a succession of rectangles that contain keywords. The GUI permits the user to avoid typing the keywords by using a system of buttons. The user can also move each keyword block and organize the script as desired.

The user can save the model in a text file. In this case, the program transforms each keyword and creates the file that will be readable for the vehicle. However, the model can be saved in a file of BIN format, which is easy to recover later. This format is just read by the GUI.

Other functions exists on the GUI in order to make sure that the generated script is conform, logical and understandable by the AUV. First, the user can judge that the entered values are right with the visualization of the commanded trajectory of the robot during the mission. Secondly, a check function permits to detect if there are some errors in the syntax of the model. Indeed, the order of the keywords is very important.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The main purpose of this work has been to develop a Graphic User Interface (GUI) to provide an easy way to program the missions. Indeed, a text file with its own syntax is used to command the mission of the robot. Thus, it took a long time to check that there was no typing errors or that the text had a logic development. The GUI is the best way to avoid all these errors that can have big consequences on the robot and its recovery.

The GUI that has been created to help the user to build the script file easily and quickly. It minimizes typing and, thus, no spelling errors can appear. Also, it gives users the ability to check if the result is correct. Indeed, special windows permit to visualize the approximation of commanded trajectory. Finally, a check function allows assuring the logic of the generated text,

This work should change the way to conduct the experiments on the AUV. Indeed, it is now easy and quick to create the script file and to be sure of the normal comportment of the robot. So, it is not useful to prepare the text file before the experiments and to take a lot of time to check the validity of the results.

B. RECOMMENDATIONS FOR FUTURE WORK

Some points haven't been achieved in the realization of this GUI. Indeed, it doesn't give the possibility to send the generated script file to the robot. After the creation of the file, the user has to use a point-to-point protocol to realize this operation. And it obviously takes time to type all the commands. So, the creation of a function that does these operations could be very useful.

Another problem appears on the function that checks the logic of the model. Having no relation with the program of the robot, it isn't prefect and doesn't check everything in the logic of the text file. For this reason, an extension of the use of the GUI could be to combine it with the Virtual World, which simulates the reaction of the robot.

Finally, as noted earlier, a previous GUI has been created [Davis 96]. This GUI is able to generate the Prolog code for the strategic level of the robot. It could be interesting to update the use of this GUI or to combine it with the new one. However, the code and the way to use the strategic level haven't yet already been defined for the new AUV. This is an important area for future work.

LIST OF REFERENCES

Brutzman, Donald P., *A Virtual World for an Autonomous Undersea Vehicle*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, December 1994.
Available at: http://www.cs.nps.navy.mil/research/auv

Marco, D. B., *Autonomous Control of Underwater Vehicles and Local Area Maneuvering*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, September 1996.
Available at http://www.cs.nps.navy.mil/research/auv

Davis, Duane, *Precision Maneuvering and Control of the Phoenix Autonomous Underwater Vehicle for Entering a Recovery Tube,* Ph.D.Dissertation, Naval Postgraduate School, Monterey, CA, December 1996.
Available at http://www.cs.nps.navy.mil/research/auv/thesispages/davis

Visual Prolog 5.0, *Getting started*, 1997.

Visual Prolog 5.0, *Language Tutorial*, 1997.

Visual Prolog 5.0, *Visual Development Environment*, 1997.

Visual Prolog 5.0, *Visual Programming Interface*, 1997.

Visual Prolog 5.0, http://www.visual-prolog.com.

APPENDIX A. MISSION SCRIPT FILE

The following appendix contains:

- 2 examples of script file (**Waypoints.script** and **TimeBased.script**)
- an explanation of all the keywords

The following script file is an example for a Waypoints control (**Waypoints.script**). The commanded trajectory for this script is as shown in **figure VII.10**.

```
TURN_ON_ADV_POWER
TURN_ON_SONAR_POWER
#
#                                  Hz    eta_z_fl  phi_z_fl  eta_psi_fl  phi_psi_fl  z_suck
GET_FLIGHT_CONTROLLER_GAINS        8     0.4       0.1       0.5         0.1         2.0
#
#                                  eta_ls  phi_ls  Km_ls  eta_rs  phi_rs  Km_rs
GET_MOTOR_CONTROLLER_GAINS         40.0    1.0     3.0    40.0    1.0     3.0
SET_MAX_DEPTH    10.0
SET_MIN_BATTERY_VOLTAGE    19.0
INITIALIZE_BOARDS
TURN_ON_PROP_POWER
ZERO_GYROS_AND_DEPTH_CELL
START_DEPTH_FILTER
WAIT  0
INITIALIZATION_DONE
#
# End Of Initialization
USE_WAYPOINT_CONTROL
#
#                          n_ls_com(Max 12.0)    n_rs_com(Max 12.0)
SET_SCREW_SPEED    12.0                          12.0
START_SCREW_SPEED_CONTROL
#                          X_com (m)  Y_com (m)  Z_com (ft)  WatchR (m)  TimeOut (Sec)
SET_WAYPOINT_XY            100.0      0.0        5.0         2.0         120.0
SET_WAYPOINT_XY            120.0      50.0       5.0         2.0         120.0
SET_WAYPOINT_XY            100.0      100.0      5.0         2.0         120.0
SET_WAYPOINT_XY            0.0        100.0      5.0         2.0         120.0
SET_WAYPOINT_XY            -20.0      50.0       0.0         2.0         120.0
SHUTDOWN
```

The following text file is an example for a Used Time Based control (**TimeBased.script**). The commanded trajectory for this script is as shown in **figure VII.11**.

```
TURN_ON_ADV_POWER
TURN_ON_SONAR_POWER
#
#                                  Hz   eta_z_fl   phi_z_fl   eta_psi_fl  phi_psi_fl  z_suck
GET_FLIGHT_CONTROLLER_GAINS   8    0.4        0.1        0.5         0.1         2.0
#
#                                  eta_ls   phi_ls   Km_ls   eta_rs   phi_rs   Km_rs
GET_MOTOR_CONTROLLER_GAINS   40.0     1.0      3.0      40.0     1.0      3.0
SET_MAX_DEPTH    10.0
SET_MIN_BATTERY_VOLTAGE    19.0
INITIALIZE_BOARDS
TURN_ON_PROP_POWER
ZERO_GYROS_AND_DEPTH_CELL
START_DEPTH_FILTER
INITIALIZATION_DONE
#
# End Of Initialization
USE_TIME_BASED_CONTROL
#
#                        n_ls_com(Max 12.0)    n_rs_com(Max 12.0)
SET_SCREW_SPEED    10.0                      10.0
START_SCREW_SPEED_CONTROL
SET_FLIGHT_HEADING    0.0
START_FLIGHT_HEADING_CONTROL
SET_FLIGHT_DEPTH    5.0
START_FLIGHT_DEPTH_CONTROL
SET_FLIGHT_DURATION    20.0
SET_FLIGHT_HEADING    180.0
SET_FLIGHT_DURATION    40.0
SURGE_CONTROL_ON
SET_FLIGHT_DURATION    20.0
SURGE_CONTROL_OFF
SET_FLIGHT_HEADING    0.0
SET_FLIGHT_DURATION    20.0
SET_FLIGHT_DEPTH    0.0
SET_FLIGHT_DURATION    5.0
SHUTDOWN
```

The following are keyword used in the script file and sent to the execution level in the robot. These are not completely consistent with the keyword nomenclature defined in [Brutzman, 94] [Davis 96] [Brutzman 98].

Initialization Primitives

TURN_ON_ADV_POWER

Unable the ADV

TURN_OFF_ADV_POWER

Enable the ADV

TURN_ON_SONAR_POWER

Unable the sonar

TURN_OFF_SONAR_POWER

Enable the sonar

GET_FLIGHT_CONTROLLER_GAINS # # # # # #

Get the gains for the flight controller
There are 6 values: Hz , eta_z_fl, hi_z_fl, eta_psi_fl, phi_psi_fl, z_suck.

GET_MOTOR_CONTROLLER_GAINS # # # # # #

Get the gains for the motor controller
There are 6 values: eta_ls, phi_ls, Km_ls, eta_rs, phi_rs, Km_rs.

SET_MAX_DEPTH #

Define the maximum depth authorized (in feet).

SET_MIN_BATTERY_VOLTAGE #

Define the minimum battery voltage authorized (in Volt).

INITIALIZE_BOARDS

Reset A/D and D/A boards.

TURN_ON_PROP_POWER	Enable Prop Servo Amplifiers.
TURN_OFF_PROP_POWER	Unable Prop Servo Amplifiers.
ZERO_GYROS_AND_DEPTH_CELL	Set current depth and heading to zero.
ZERO _DEPTH_CELL	Set current depth to zero.
START_DEPTH_FILTER	Start the filter that smoothes the signal from the Depth Cell.
IGNORE_LEAK_CHECK	Ignore the apparition of a leak on the boat and continue the mission.
IGNORE_VOLTAGE_CHECK	Continue the mission even though the voltage power is weak.
SET_GPS_ORIGIN # #	Attribute the origin of the GPS coordinates (Longitude and Latitude in milliseconds)
WAIT #	Wait the period of time specified (in Sec).
INITIALIZATION_DONE	Declare the end of the Initialization section.

Control Primitives

USE_WAYPOINT_CONTROL

Specify that the robot is going to use the waypoint control. It consists to send the robot to different points.

USE_TIME_BASED_CONTROL

Specify that the robot is going to use the time based control. It consists to guide the robot with heading and time.

SET_SCREW_SPEED # #

Set the speed of the left and right propellers

SET_SCREW_SPEED_FROM_FILE

Obtain the speed of the propellers from a file. This is used for System Identification.

START_SCREW_SPEED_CONTROL

Start the control of the screw speed with the speed done in "SET_SCREW_SPEED" or "SET_SCREW_SPEED_FROM_FILE".

STOP_SCREW_SPEED_CONTROL

Stop the control of the screw speed.

SET_WAYPOINT_XY # # # # #

Specify a point where the robot have to go. It indicates the coordonates X, Y (in meters) and the depth Z (in feet). The fourth value is a margin, which is a sphere of radius R (in meters), and the fifth one indicates the time let to the robot to join the point (in sec).

SET_WAYPOINT_GPS # # # # #

Specify a point where the robot have to go using the GPS coordinates. The two first values indicates the longitude and latitude in milliseconds. The three others are the same as in SET_WAYPOINT_XY.

SET_FLIGHT_HEADING #	Enter the flight heading (in deg).
START_FLIGHT_HEADING_CONTROL	Start the flight heading control to keep the last heading specified.
STOP_FLIGHT_HEADING_CONTROL	Stop the flight heading control.
SET_FLIGHT_DEPTH #	Enter the flight depth (in feet).
START_FLIGHT_DEPTH_CONTROL	Start the flight depth control to keep the last depth specified.
STOP_FLIGHT_DEPTH_CONTROL	Stop the flight depth control.
SET_FLIGHT_DURATION #	Define the time for the flight (in sec)
START_DEPTH_ERROR_FILTER	Enable the depth error filter, which makes sure to obtain the specified depth.
START_HEADING_ERROR_FILTER	Enable the heading error filter, which makes sure to obtain the specified heading.
SURGE_CONTROL_ON	Enable the surge control which permits to keep a position on the surge direction (x).
SURGE_CONTROL_OFF	Unable the surge control.
HEADING_AND_SWAY_CONTROL # #	Get the values of the heading and sway direction (y).

SUBMERGE # # #	Use the thrusters to submerge the robot to a specified depth.
ROTATE # # #	Use the thrusters to turn the robot to a specified heading.
SURFACE	Order the robot to surface.
SET_SCREW_VOLTAGE #	Set the voltage that is send to the propellers.
START_SCREW_VOLTAGE_CONTROL	Start the control of the propellers with the voltage command.
SET_FIXED_PLANE_ANGLES # #	Set the Plane and Rudder angles values for the fins.
START_FIXED_PLANE_CONTROL	Start the control of the fins with the Plane and Rudder angles command.
SHUTDOWN	End of the mission.

APPENDIX B. CODE LISTING

The following is a code listing of all the Prolog programs that have been created. There is a description of the different programs:

- **Script.pre**: declaration of the global predicates, 1 page
- **Script.pro**: main program, 27 pages
- **Vpitools.pro**: insertion of Visual Prolog tools, 2 pages
- **Manag_commands.pro**: Managing the main menu and the buttons, 5 pages
- **Dialogs.pro**: Managing the dialogs boxes, 19 pages
- **Waypoints.pro**: Visualizing the trajectory in Waypoints control, 6 pages
- **Time_based.pro**: Visualizing the trajectory in Time based control, 7 pages
- **Check_all_model.pro**: Check the logic of the model, 5 pages

```
/*****************************************************************************

            Copyright (c) NPS

 Project:  SCRIPT
 FileName: SCRIPT.PRE
 Purpose: Predicate definitions for SCRIPT.PRO
 Written by: Joel Doleac
 Comments:
*****************************************************************************/

%BEGIN_DECL, System generated global predicates
GLOBAL PREDICATES

  project_ShowHelpContext(INTEGER Index) - (i)
  dlg_about_dialog_Create(WINDOW Parent) - (i)
  tb_project_toolbar_Create(WINDOW Parent) - (i)
  tb_help_line_Create(WINDOW Parent) - (i)
  tb_script_Create(WINDOW Parent) - (i)


  tb_script1_Create(WINDOW Parent) - (i)
  tb_script2_Create(WINDOW Parent) - (i)
%END_DECL
  win_waypoints_Create(WINDOW)
  %win_time_based_flights_Create(WINDOW)

  set_enabled_menu(WINDOW)
  set_main_menu(WINDOW,BOOLEAN)
  set_initialization_toolbar(WINDOW,BOOLEAN)
  set_waypoint_control_Toolbar(WINDOW,BOOLEAN)
  set_time_based_control_Toolbar(WINDOW,BOOLEAN)
  set_control_common_Toolbar(WINDOW,BOOLEAN)
  nondeterm noted_toolbar(WINDOW,INTEGER,STRING)
  determin_text(posit,Integer,posit,text,String) - (i,i,i,i,o)
  otherexist(posit,Integer,text,String) - (i,i,i,o)


  check_right_model
```

```
/*****************************************************************************

            Copyright (c) NPS

 Project:  SCRIPT
 FileName: SCRIPT.PRO
 Purpose: Generation of a script file
 Written by: Joel Doleac
 Comments: This program is the main program of the GUI. The purpose of this GUI
           is to make easier the generation of a text file. It is used by the
           NPS center for AUV research to define the mission of the AUV.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"


Database - keepFile
    determ keep_file(String)


/*****************************************************************************
                    Redraw the drawing
  Each time you change something in the window, you call this function
*****************************************************************************/

PREDICATES
    ligne(WINDOW,integer,integer,integer)
    drawing_with_values(WINDOW,values)
    dessin(WINDOW,INTEGER)
    winRefresh(WINDOW)



CLAUSES
   winRefresh(_Win):-
       Rct = win_GetClip(_Win),
       win_Clear(_Win,Rct,color_White),
       counter(Num),
       Max=60*Num+90,
       win_SetScrollRange(_Win,sb_Vert,0,Max),
       RCT = win_GetClip(_Win),
       RCT = rct(_,_,_,B),
       win_SetScrollProportion(_Win, sb_Vert, B),
       dessin(_Win,1).


   dessin(_Win,Posit):-
       counter(Num),
       Posit<>Num+1,
       keyword(Posit,message(Text,Values),Color),!,
       noted_toolbar(_Win,Posit,Text),
       PosScroll = win_GetScrollPos(_Win,sb_Vert),
       Top = 60*Posit-10-PosScroll,
       Bottom = Top+50,
       RCTwin = win_GetClip(_Win),
       RCTwin = rct(L,_,R,_),
       Middle = (R+L) div 2,
       ligne(_Win,Posit,Top,Middle),
       Right = Middle-150, Left = Middle+150,
       RCT = rct(Right,Top,Left,Bottom),
       win_SetBrush(_Win,brush(pat_Solid,Color)),
```

```
        drawing_with_values(_Win,Values),
        draw_Rect(_Win,RCT),
        determin_text(1,1,Posit,Text,NewText),
        draw_TextInRect(_Win, RCT, NewText, -1,[dtext_center,dtext_vcenter,
        dtext_singleline]),
        XPos = Right - 40,
        YPos = Top + 30,
        str_int (StPosit, Posit),
        draw_Text (_Win, XPos, YPos, StPosit),
        NewPosit=Posit+1,
        dessin(_Win,NewPosit),
        !.
    dessin(_,_):-!.

    ligne(_Win,1,_,_):-!.

    ligne(_Win,_,Top,Middle):-
        X=Top-11,
        Pen = pen(1 , ps_Solid, color_Black),
        win_SetPen(_Win, Pen),
        draw_Line(_Win,pnt(Middle,Top),pnt(Middle,X)),
        !.

    drawing_with_values(_Win,[]):-!,
        Pen = pen(1 , ps_Solid, color_Black),
        win_SetPen(_Win, Pen).
    drawing_with_values(_Win,[r(0.0)]):-!,
        Pen = pen(3 , ps_Solid, color_Red),
        win_SetPen(_Win, Pen).
    drawing_with_values(_Win,[r(0.0)|T]):-!,
        drawing_with_values(_Win,T).
    drawing_with_values(_Win,_):-!,
        Pen = pen(2 , ps_Solid, color_Blue),
        win_SetPen(_Win, Pen).


    determin_text(P,NbInf,Posit,Text,NewText):-
        P<Posit,
        keyword(P,message(Text,_),_),!,
        NewNbInf = NbInf+1,
        NewP = P+1,
        determin_text(NewP,NewNbInf,Posit,Text,NewText).
    determin_text(P,NbInf,Posit,Text,NewText):-
        P<Posit,!,
        NewP = P+1,
        determin_text(NewP,NbInf,Posit,Text,NewText).
    determin_text(P,NbInf,Posit,Text,NewText):-
        P = Posit,!,
        NewP=P+1,
        otherexist(NewP,NbInf,Text,NewText).

    otherexist(P,1,Text,NewText):-
        keyword(P,message(Text,_),_),!,
        concat(Text," 1",NewText).
    otherexist(P,1,Text,NewText):-
        counter(Num),
        P<>Num+1,!,
        NewP = P+1,
        otherexist(NewP,1,Text,NewText).
    otherexist(_,1,Text,NewText):-!,
        NewText = Text.
    otherexist(_,NbInf,Text,NewText):-
        str_Int(StNb,NbInf),
```

```
        concat(" ",StNb,Text1),
        concat(Text,Text1,NewText).


/*****************************************************************************
                        Use of the gray rectangle
*****************************************************************************/
predicates
    delete_LtGray
    rectangle_below(WINDOW,PNT,posit) - (i,i,o)
    move_GrayCase(WINDOW,Posit)
    move_scrollBarWithGray(WINDOW,Posit)

clauses
   delete_LtGray:-
       retract(keyword(Posit,message(Text,Values),color_LtGray)),!,
       assert(keyword(Posit,message(Text,Values),color_White)).
   delete_LtGray:-!.

   rectangle_below(_Win,PNT,Posit):-
       keyword(Posit,_,_),
       PosScroll = win_GetScrollPos(_Win,sb_Vert),
       Top = 60*Posit-10-PosScroll,
       Bottom = Top+50,
       RCT=rct(350,Top,650,Bottom),
       rect_PntInside(RCT,PNT),!,
       retract(keyword(Posit,message(Text,Values),_)),!,
       assert(keyword(Posit,message(Text,Values),color_LtGray)).
   rectangle_below(_,_,Num):-!,
       counter(Num),
       retract(keyword(Num,Messages,_)),!,
       assert(keyword(Num,Messages,color_LtGray)).


   move_GrayCase(_,0):-!.
   move_GrayCase(_,Max):-
        counter(Num),
        Max = Num +1,!.
   move_GrayCase(Win,NewPosit):-
       retract(keyword(Posit,Message,color_LtGray)),!,
       assert(keyword(Posit,Message,color_White)),
       retract(keyword(NewPosit,LastMessage,_)),!,
       assert(keyword(NewPosit,LastMessage,color_LtGray)),
       move_scrollBarWithGray(Win,NewPosit).

   move_scrollBarWithGray(Win,Posit):-
       Rct = win_GetClip(Win),
       Rct = rct(_,_,_,Bottom),
       PosScroll = win_GetScrollPos(Win,sb_Vert),
       Top = 60*Posit-10-PosScroll,
       Top > Bottom-30,!,
       NewPos = PosScroll+60,
       win_SetScrollPos(Win,sb_Vert, NewPos).
   move_scrollBarWithGray(Win,Posit):-
       Rct = win_GetClip(Win),
       Rct = rct(_,T,_,_),
       PosScroll = win_GetScrollPos(Win,sb_Vert),
       Top = 60*Posit-10-PosScroll,
       Top < T,!,
       NewPos = PosScroll-60,
       win_SetScrollPos(Win,sb_Vert, NewPos).
   move_scrollBarWithGray(_,_):-!.
```

```
/*****************************************************************************
                                Decide toolbars
 *****************************************************************************/
 predicates
   decide_toolbar(WINDOW)

 clauses
   decide_toolbar(Win):-
       keyword(_,message("USE WAYPOINT CONTROL",_),_),!,
       set_initialization_toolbar(Win,1),
       set_time_based_control_Toolbar(Win,0),
       set_waypoint_control_Toolbar(Win,1),
       set_control_common_Toolbar(Win,1),
       set_main_menu(Win,1).
   decide_toolbar(Win):-
       keyword(_,message("USE TIME BASED CONTROL",_),_),!,
       set_initialization_toolbar(Win,1),
       set_waypoint_control_Toolbar(Win,0),
       set_time_based_control_Toolbar(Win,1),
       set_control_common_Toolbar(Win,1),
       set_main_menu(Win,1).
   decide_toolbar(Win):-
       keyword(1,_,_),!,
       set_control_common_Toolbar(Win,0),
       set_initialization_toolbar(Win,1),
       set_time_based_control_Toolbar(Win,0),
       set_waypoint_control_Toolbar(Win,0),
       set_main_menu(Win,1).
   decide_toolbar(Win):-
       TitleWin = win_GetText (Win), TitleWin <> "script",!,
       set_control_common_Toolbar(Win,0),
       set_initialization_toolbar(Win,1),
       set_time_based_control_Toolbar(Win,0),
       set_waypoint_control_Toolbar(Win,0),
       set_main_menu(Win,1).
   decide_toolbar(Win):-!,
       set_initialization_toolbar(Win,0),
       set_time_based_control_Toolbar(Win,0),
       set_waypoint_control_Toolbar(Win,0),
       set_control_common_Toolbar(Win,0),
       set_main_menu(Win,0).

/*****************************************************************************
                    Manipulation of the "keywords" database
*****************************************************************************/
predicates
   build_keyword(WINDOW,String)
   paste_keyword(WINDOW)
   check_already_exist(WINDOW,String)
   delete_databases
   position_GrayRct(Integer) - (o)
   loop_increase(posit,messages,Color)
   delete_rectangle(WINDOW)
   number_to_delete(posit,text)
   grayed_previous(Integer)
   delete_next(posit)
   loop_decrease(posit,Color)
   suite(posit)
   check_text_with_values(text,values) - (i,o)
```

```
clauses
  build_keyword(_Win,Text):-      %create the keyword associate with Text in the database
       retract(counter(Num)),
       NewNum = Num+1,
       assert(counter(NewNum)),
       position_GrayRct(Posit),
       check_text_with_values(Text,Values),
       assert(keyword(Posit,message(Text,Values),color_Ltgray)),
       move_scrollBarWithGray(_Win,Posit),
       winRefresh(_Win),!.


  position_GrayRct(Posit):-                    %If a gray Rectangle exist, this function
       retract(keyword(GrayPosit,GrayMessages,color_LtGray)),!,     %put the next
                                                                    rectangle after this one.
       assert(keyword(GrayPosit,GrayMessages,color_White)),
       Posit=GrayPosit+1,
       suite(Posit).
  position_GrayRct(Num):-
       counter(Num),!.

  suite(Num):-
       counter(Num),!.
  suite(Posit):-
       retract(keyword(Posit,Messages,Color)),!,
       NextPosit=Posit+1,
       loop_increase(NextPosit,Messages,Color).


  loop_increase(Posit,LastMessages,LastColor):-       %move each rectangle after
       retract(keyword(Posit,Messages,Color)),!,      %the added rectangle.
       assert(keyword(Posit,LastMessages,LastColor)),
       NextPosit=Posit+1,
       loop_increase(NextPosit,Messages,Color).
  loop_increase(NextPosit,Messages,Color):-
       assert(keyword(NextPosit,Messages,Color)),!.


  check_text_with_values("Get flight controller gains",Values):-!, %Add values if
                                                                   %necessary
       Values = [r(8.0),r(0.4),r(0.1),r(0.5),r(0.1),r(2.0)].
  check_text_with_values("Get motor controller gains",Values):-!,
       Values = [r(40.0),r(1.0),r(3.0),r(40.0),r(1.0),r(3.0)].
  check_text_with_values("Set max depth",Values):-!,
       Values = [r(10.0)].
  check_text_with_values("Set min battery voltage",Values):-!,
       Values = [r(19.0)].
  check_text_with_values("Set GPS origin",Values):-!,
       Values = [r(0.0),r(0.0)].
  check_text_with_values("Wait",Values):-!,
       Values = [r(0.0)].
  check_text_with_values("Set screw speed",Values):-!,
       Values = [r(0.0),r(0.0)].
  check_text_with_values("Set flight heading",Values):-!,
       Values = [r(0.0)].
  check_text_with_values("Set flight depth",Values):-!,
       Values = [r(0.0)].
  check_text_with_values("Set flight duration",Values):-!,
       Values = [r(0.0)].
  check_text_with_values("Set waypoint XY",Values):-!,
       Values = [r(0.0),r(0.0),r(0.0),r(0.0),r(0.0)].
  check_text_with_values("Set waypoint GPS",Values):-!,
       Values = [r(0.0),r(0.0),r(0.0),r(0.0),r(0.0)].
```

```
check_text_with_values("Heading and sway control",Values):-!,
     Values = [r(0.0),r(0.0)].
check_text_with_values("Submerge",Values):-!,
     Values = [r(0.0),r(0.0),r(0.0)].
check_text_with_values("Rotate",Values):-!,
     Values = [r(0.0),r(0.0),r(0.0)].
check_text_with_values("Set screw voltage",Values):-!,
     Values = [r(0.0)].
check_text_with_values("Set fixed plane angles",Values):-!,
     Values = [r(0.0),r(0.0)].
check_text_with_values(_,Values):-!,
     Values = [].


paste_keyword(_Win):-                          %Paste a keyword after the gray rectangle
     Bin = cb_GetBin("Message"),
     term_bin(messages,message(Text,Values),Bin),
     retract(counter(Num)),
     NewNum = Num+1,
     assert(counter(NewNum)),
     position_GrayRct(Posit),
     assert(keyword(Posit,message(Text,Values),color_LtGray)),
     winRefresh(_Win).


delete_rectangle(_Win):-                       %Delete a keyword
     retract(keyword(GrayPosit,message(Text,_),color_LtGray)),!,
     number_to_delete(GrayPosit,Text),
     decide_toolbar(_Win),
     winRefresh(_Win).
delete_rectangle(_Win):-!,winRefresh(_Win).

number_to_delete(GrayPosit,"USE WAYPOINT CONTROL"):-!,
     retract(counter(_)),
     NewNum = GrayPosit-1,
     assert(counter(NewNum)),
     grayed_previous(NewNum),
     delete_next(GrayPosit).
number_to_delete(GrayPosit,"USE TIME BASED CONTROL"):-!,
     retract(counter(_)),
     NewNum = GrayPosit-1,
     assert(counter(NewNum)),
     grayed_previous(NewNum),
     delete_next(GrayPosit).
number_to_delete(GrayPosit,_):-!,
     retract(counter(Num)),
     NewNum = Num-1,
     assert(counter(NewNum)),
     loop_decrease(GrayPosit,color_LtGray).

grayed_previous(0):-!.
grayed_previous(Num):-
     retract(keyword(Num,Messages,_)),!,
     assert(keyword(Num,Messages,color_LtGray)).

delete_next(Posit):-
     NewPosit = Posit+1,
     retract(keyword(NewPosit,_,_)),!,
     delete_next(NewPosit).
delete_next(_).
```

```
  loop_decrease(Posit,Color):-
       NextPosit = Posit+1,
       retract(keyword(NextPosit,Messages,NextColor)),!,
       assert(keyword(Posit,Messages,Color)),
       loop_decrease(NextPosit,NextColor).
  loop_decrease(_,Color):-
       Color<>color_White,
       counter(Num),
       retract(keyword(Num,Messages,_)),!,
       assert(keyword(Num,Messages,Color)).
  loop_decrease(_,_):-!.


  check_already_exist(_Win,Text):-      %check if the keywords "WAYPOINT CONTROL" or
       keyword(_,message(Text,_),_),!.  %"TIME BASED CONTROL" exist already in the
                                        database
  check_already_exist(_Win,Text):-
       build_keyword(_Win,Text),!.


  delete_databases:-                    %Initialize all the databases
       retractall(_,keywords),
       retractall(_,keepFile).


/*************************************************************************
         Call the dialog boxes in order to change the values
*************************************************************************/
predicates
   values_associated(WINDOW,posit,text)

clauses
   values_associated(_Win,Posit,"Get flight controller gains"):-!,
       dlg_get_flight_controller_gains_Create(_Win,Posit).
   values_associated(_Win,Posit,"Get motor controller gains"):-!,
       dlg_get_motor_controller_gains_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set max depth"):-!,
       dlg_set_max_depth_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set min battery voltage"):-!,
       dlg_set_min_battery_voltage_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set GPS origin"):-!,
       dlg_set_gps_origin_Create(_Win,Posit).
   values_associated(_Win,Posit,"Wait"):-!,
       dlg_wait_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set screw speed"):-!,
       dlg_set_screw_speed_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set waypoint XY"):-!,
       dlg_set_waypoint_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set waypoint GPS"):-!,
       dlg_set_waypoint_gps_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set flight heading"):-!,
       dlg_set_flight_heading_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set flight depth"):-!,
       dlg_set_flight_depth_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set flight duration"):-!,
       dlg_set_flight_duration_Create(_Win,Posit).
   values_associated(_Win,Posit,"Heading and sway control"):-!,
       dlg_heading_and_sway_control_Create(_Win,Posit).
   values_associated(_Win,Posit,"Submerge"):-!,
       dlg_submerge_Create(_Win,Posit).
   values_associated(_Win,Posit,"Rotate"):-!,
       dlg_rotate_Create(_Win,Posit).
```

```
   values_associated(_Win,Posit,"Set screw voltage"):-!,
      dlg_set_screw_voltage_Create(_Win,Posit).
   values_associated(_Win,Posit,"Set fixed plane angles"):-!,
      dlg_set_fixed_plane_angles_Create(_Win,Posit).

/*****************************************************************************
              Use to open or create a new file to put the model
*****************************************************************************/
predicates
    check_exist_file(WINDOW, String)
    create_file(WINDOW,String)
    open_file(WINDOW,String)

clauses
   check_exist_file(_Win,Filename):-
      existfile(Filename),!,
      open_file(_Win,Filename).
   check_exist_file(_Win,Filename):-
      create_file(_Win,Filename),!.

   open_file(_Win,Filename):-     %Open a file which already exist, put the values in
      filenamepath(Filename,_,Title),  %the database and save its name in another
                                       database
      win_SetText(_Win, Title),
      consult(Filename,keywords),
      decide_toolbar(_Win),
      assert(keep_file(Filename)),
      !.

   create_file(_Win,Filename):-  %Inialilize the counter and save a name for a new file
      filenamepath(Filename,_,Title),
      win_SetText(_Win, Title),
      assert(keep_file(Filename)),
      assert(counter(0)),
      decide_toolbar(_Win).

/*****************************************************************************
             Decide if a file as to be saved before closing
*****************************************************************************/
predicates
   interpret(WINDOW,Integer)
   yesOrNo(WINDOW,Integer)
   change_filename(WINDOW,String,String)
   no_keyword(WINDOW)

clauses
                          %use to define if there is something in the database
  no_keyword(Win):-       %before opening another file
      keyword(1,_,_),!,
      Response  = dlg_Ask ("Do you want to save the model?", ["Yes","No","Cancel"]),
      interpret(Win,Response).
  no_keyword(_).


  interpret(_,resp_3):-!.              %interpret the answer to the question:
  interpret(_Win,OtherResp):-          %"Do you want to save the model?"
      yesOrNo(_Win,OtherResp),
      win_SetText(_Win,"script"),
      delete_databases,
      decide_toolbar(_Win).
```

```
  yesOrNo(_,resp_2):-!.
  yesOrNo(Win,resp_default):-!,
       keep_file(Filename),
       filenamepath(Filename,_,Title),
       change_filename(Win,Title,Filename).


/***************************************************************************
          Check if the title of the file is not "Untitled.bin"
          or "script.d" before saving
***************************************************************************/


clauses
   change_filename(_Win,Titled,Filename):-
       filenameext(Titled,_,Ext),
       Ext = ".d",!,
       NewFilename = dlg_GetFileName( "*.bin", ["Bin files","*.bin","All files","*.*"],
                "Save as BIN file ", [dlgfn_Save], "", _),
       retract(keep_file(Filename)),
       save(NewFilename,keywords).
   change_filename(_Win,"Untitled.bin",Filename):-!,
       NewFilename = dlg_GetFileName( "*.bin", ["Bin files","*.bin","All files","*.*"],
                "Save as BIN file ", [dlgfn_Save], "", _),
       retract(keep_file(Filename)),
       assert(keep_file(NewFilename)),
       filenamepath(NewFilename,_,Title),
       win_SetText(_Win,Title),
       save(NewFilename,keywords).
   change_filename(_,_,Filename):-
       save(Filename,keywords),!.

 /***************************************************************************
                  Write the database into the script file
 ***************************************************************************/

 predicates
   loop_save(Integer)
   write_values(values,text)
   write_a_list(values,String,Unsigned,text)
   comment(Integer,String,posit)
   compose_key(String)

 clauses
   loop_save(Posit):-
       keyword(Posit,message(Text,Values),_),!,
       upper_lower(Word,Text),
       comment(1,Text,Posit),
       compose_key(Word),
       write_values(Values,Text),nl,
       comment(2,Text,Posit),
       NewPosit = Posit+1,
       loop_save(NewPosit).

   loop_save(_):-!.

   comment(1,"Get flight controller gains",_):-!,
       write("#\n#                                Hz    eta_z_fl    phi_z_fl
eta_psi_fl",
       "  phi_psi_fl  z_suck"),nl.

   comment(1,"Get motor controller gains",_):-
```

```
      write("#\n#                                    eta_ls     phi_ls     Km_ls      eta_rs",
      "     phi_rs     Km_rs"),nl,!.

   comment(1,"Set screw speed",_):-
      write("#\n#                    n_ls_com(Max 12.0)     n_rs_com(Max 12.0)"),nl,!.

   comment(1,"Set waypoint XY",Posit):-
      determin_text(1,1,Posit,"Set waypoint XY",NewText),
      NewText = "Set waypoint XY",!,
      write("#                    X_com (m)  Y_com (m)  Z_com (ft) WatchR (m) TimeOut
(Sec)"),nl.

   comment(1,"Set waypoint XY",Posit):-
      determin_text(1,1,Posit,"Set waypoint XY",NewText),
      NewText ="Set waypoint XY 1",!,
      write("#                    X_com (m)  Y_com (m)  Z_com (ft) WatchR (m) TimeOut
(Sec)"),nl.

   comment(1,"Set waypoint GPS",Posit):-
      determin_text(1,1,Posit,"Set waypoint GPS",NewText),
      NewText = "Set waypoint GPS",!,
      write("#                    Long (msec)    Lat (msec)     Z_com (ft)     WatchR (m)
TimeOut (Sec)"),nl.

   comment(1,"Set waypoint GPS",Posit):-
      determin_text(1,1,Posit,"Set waypoint GPS",NewText),
      NewText ="Set waypoint GPS 1",!,
      write("#                    Long (msec)    Lat (msec)     Z_com (ft)     WatchR (m)
TimeOut (Sec)"),nl.

   comment(2,"Initialization done",_):-
      write("#\n# End Of Initialization"),nl,!.

   comment(_,_,_):-!.

   compose_key(Word):-
      searchstring(Word," ",Pos),!,
      Num=Pos-1,
      frontstr(Num,Word,Str1,Str2),
      concat(Str1,"_",NewWord),
      write(NewWord),
      frontstr(1,Str2,_,Rest),
      compose_key(Rest).
   compose_key(Rest):-
      write(Rest).

   write_values([r(X)],"Wait"):-!,
      write("    ",X," ").
   write_values([H|T],"Get flight controller gains"):-!,
      H = r(X),
      write("     ",X),
      write_a_list(T,"     ",7,"Get flight controller gains").
   write_values(Values,"Set screw speed"):-!,
      write_a_list(Values,"     ",18,"Set screw speed").
   write_values(Values,"Set waypoint GPS"):-!,
      write_a_list(Values,"    ",12,"Set waypoint GPS").
   write_values(Values,"Set GPS origin"):-!,
      write_a_list(Values,"  ",12,"Set GPS origin").
   write_values(Values,Text):-
      write_a_list(Values,"     ",6,Text).

   write_a_list([],_,_,_).
   write_a_list([r(X)],Space,_,"Submerge"):-!,
```

```
        write(Space,X).
    write_a_list([r(X)],Space,_,"Rotate"):-!,
        write(Space,X).
    write_a_list([H|T],Space,Delta,Text):-          /* Match the head to H and the tail to
                                                    T, then... */
        H = r(X),
        str_real(St,X),
        searchchar(St,'.',_),!,
        str_len (St, Lenght),
        Nspace = Delta - Lenght,
        str_len(StAfter,Nspace),
        write(Space,X,StAfter),
        write_a_list(T,Space,Delta,Text).
    write_a_list([H|T],Space,Delta,Text):-!,
        H = r(X),
        str_real(St,X),
        str_len (St, Lenght),
        Nspace = Delta - Lenght - 2,
        str_len(StAfter,Nspace),
        write(Space,X,".0",StAfter),
        write_a_list(T,Space,Delta,Text).

/*****************************************************************************
              Read the script file and put it into the Database
*****************************************************************************/

PREDICATES

  readline(integer)
  decompose_line(string)
  decompose_key(string,string)
  decompose_number(string)
  add_value(values,values,Real)

CLAUSES
  readline(Nomb):-
        readdevice(script),
        not(eof(script)),!,
        readln(Text),
        readdevice(_),
        decompose_line(Text),
        readline(Nomb).
  readline(_):-
        counter(Num),
        retract(keyword(Num,Messages,_)),!,
        assert(keyword(Num,Messages,color_LtGray)).

    decompose_line(Text):-
        subchar(Text,1,Ch),
        Ch = '#',!.
    decompose_line(Text):-
        searchchar(Text,' ',Pos),!,
        LastPos=Pos-1,
        frontstr(LastPos,Text,Str1,Str2),
        frontstr(1,Str1,First,Rest),
        upper_lower(Rest,Low),
        retract(counter(Num)),
        NewNum = Num + 1,
        assert(counter(NewNum)),
        decompose_key(Low,First),
        decompose_number(Str2).
    decompose_line(Rest):-!,
        retract(counter(Num)),
```

```
        NewNum = Num + 1,
        assert(counter(NewNum)),
        frontstr(1,Rest,First,Str),
        upper_lower(Str,Word),
        decompose_key(Word,First).

    decompose_key("se_waypoint_control","U"):-!,
        counter(Num),
        assert(keyword(Num,message("USE WAYPOINT CONTROL",[]),color_white)).
    decompose_key("se_time_based_control","U"):-!,
        counter(Num),
        assert(keyword(Num,message("USE TIME BASED CONTROL",[]),color_white)).
    decompose_key("urn_on_adv_power","T"):-!,
        counter(Num),
        assert(keyword(Num,message("Turn on ADV power",[]),color_white)).
    decompose_key("urn_off_adv_power","T"):-!,
        counter(Num),
        assert(keyword(Num,message("Turn off ADV power",[]),color_white)).
    decompose_key("et_waypoint_xy","S"):-!,
        counter(Num),
        assert(keyword(Num,message("Set waypoint XY",[]),color_white)).
    decompose_key("et_waypoint_gps","S"):-!,
        counter(Num),
        assert(keyword(Num,message("Set waypoint GPS",[]),color_white)).
    decompose_key("et_gps_origin","S"):-!,
        counter(Num),
        assert(keyword(Num,message("Set GPS origin",[]),color_white)).
    decompose_key(Key,Text):-
        searchstring(Key,"_",Pos),!,
        Num = Pos-1,
        frontstr(Num,Key,Str1,Str2),
        concat(Str1," ",Text2),
        concat(Text,Text2,NewText),
        frontstr(1,Str2,_,Rest),
        decompose_key(Rest,NewText).
    decompose_key(LastWord,Text):-!,
        concat(Text,LastWord,FinalText),
        counter(Num),
        assert(keyword(Num,message(FinalText,[]),color_white)).


    decompose_number(""):-!.
    decompose_number(Text):-
        subchar(Text,1,Ch),
        Ch = ' ',!,
        frontstr(1,Text,_,Rest),
        decompose_number(Rest).
    decompose_number(Rest):-
        searchchar(Rest,' ',Pos),!,
        Num=Pos-1,
        frontstr(Num,Rest,Str1,Str2),
        str_real(Str1,Real),
        counter(Posit),
        retract(keyword(Posit,message(Text,OldValues),Color)),!,
        add_value(OldValues,Values,Real),
        assert(keyword(Posit,message(Text,Values),Color)),
        decompose_number(Str2).
    decompose_number(Rest):-!,
        str_real(Rest,Real),
        counter(Posit),
        retract(keyword(Posit,message(Text,OldValues),Color)),!,
        add_value(OldValues,Values,Real),
        assert(keyword(Posit,message(Text,Values),Color)).
```

```
   add_value([],[r(Real)],Real):-!.
   add_value([Head],[Head,r(Real)],Real):-!.
   add_value([Head|Tail],[Head|NewTail],Real):-!,
      add_value(Tail,NewTail,Real).


%BEGIN_WIN Task Window
/*****************************************************************************
              Event handling for Task Window
*****************************************************************************/

predicates

  task_win_eh : EHANDLER

constants

%BEGIN Task Window, CreateParms, 15:03:57-22.6.1999, Code automatically updated!
  task_win_Flags =
[wsf_SizeBorder,wsf_TitleBar,wsf_Close,wsf_Maximize,wsf_Minimize,wsf_ClipSiblings,wsf_Ma
ximized,wsf_VScroll]
  task_win_Menu  = res_menu(idr_task_menu)
  task_win_Title = "script"
  task_win_Help  = idh_contents
%END Task Window, CreateParms

clauses

%BEGIN Task Window, e_Create
  task_win_eh(_Win,e_Create(_),0):-!,
%BEGIN Task Window, InitControls, 15:03:57-22.6.1999, Code automatically updated!
%END Task Window, InitControls
%BEGIN Task Window, ToolbarCreate, 15:03:57-22.6.1999, Code automatically updated!
      tb_script1_Create(_Win),
      tb_script_Create(_Win),
      tb_project_toolbar_Create(_Win),
%END Task Window, ToolbarCreate
ifdef use_message
      msg_Create(100),
enddef
      !.
%END Task Window, e_Create

%MARK Task Window, new events

%BEGIN Task Window, id_file_new
  task_win_eh(_Win,e_Menu(id_file_new,_ShiftCtlAlt),0):-!,
      no_keyword(_Win),
      disk(Path),
      format(Filename,"%s\\Untitled.bin",Path),
      delete_databases,
      set_enabled_menu(_Win),
      check_exist_file(_Win,Filename),
      winRefresh(_Win),
      !.
%END Task Window, id_file_new

%BEGIN Task Window, id_file_open
  task_win_eh(_Win,e_Menu(id_file_open,_ShiftCtlAlt),0):-!,
      no_keyword(_Win),
      FileName = dlg_GetFileName( "Untitled.bin", ["Bin files","*.bin","All
files","*.*"],
```

```
            "Choose BIN file ", [], "", _),
      delete_databases,
      set_enabled_menu(_Win),
      check_exist_file(_Win, Filename),
      winRefresh(_Win),
      !.
%END Task Window, id_file_open

%BEGIN Task Window, id_file_close
  task_win_eh(_Win,e_Menu(id_file_close,_ShiftCtlAlt),0):-!,
      Response  = dlg_Ask ("Do you want to save the model?", ["Yes","No","Cancel"]),
      interpret(_Win,Response),
      winRefresh(_Win),
      !.
%END Task Window, id_file_close

%BEGIN Task Window, id_file_save_model
  task_win_eh(_Win,e_Menu(id_file_save_model,_ShiftCtlAlt),0):-!,
      keep_file(Filename),
      filenamepath(Filename,_,Title),
      change_filename(_Win,Title,Filename),
      !.
%END Task Window, id_file_save_model

%BEGIN Task Window, id_file_save_as
  task_win_eh(_Win,e_Menu(id_file_save_as,_ShiftCtlAlt),0):-!,
      Newname = dlg_GetFileName( "*.bin", ["Bin files","*.bin","All files","*.*"],
               "Save as BIN file ", [dlgfn_Save], "", _),
      save(Newname,keywords),
      !.
%END Task Window, id_file_save_as

%BEGIN Task Window, id_Script_openScript
  task_win_eh(_Win,e_Menu(id_Script_openScript,_ShiftCtlAlt),0):-!,
      no_keyword(_Win),
      FileName = dlg_GetFileName( "script", ["script files","*.d"],
                "Choose script file ", [], "", _),
      delete_databases,
      assert(keep_file(Filename)),
      openread(script, Filename),
      readdevice(_),
      assert(counter(0)),
      readline(1),
      closefile(script),
      set_enabled_menu(_Win),
      filenamepath(Filename,_,Title),
      win_SetText(_Win, Title),
      decide_toolbar(_Win),
      winRefresh(_Win),
      !.
%END Task Window, id_Script_openScript

%BEGIN Task Window, id_Script_saveScript
  task_win_eh(_Win,e_Menu(id_Script_saveScript,_ShiftCtlAlt),0):-!,
      Filename = dlg_GetFileName( "script.d", ["script files","*.d"],
               "Save as script file ", [dlgfn_Save], "", _),
      openwrite(script,Filename),
      writedevice(script),
      loop_save(1),
      closefile(script),
      dlg_Note("Save in the Script","The Script has been built."),
      !.
%END Task Window, id_Script_saveScript
```

```
%BEGIN Task Window, id_file_exit
  task_win_eh(Win,e_Menu(id_file_exit,_ShiftCtlAlt),0):-!,
       win_Destroy(Win),
       !.
%END Task Window, id_file_exit

%BEGIN Task Window, id_Display_waypoints
  task_win_eh(_Win,e_Menu(id_Display_waypoints,_ShiftCtlAlt),0):-!,
       win_waypoints_Create(_Win),
       !.
%END Task Window, id_Display_waypoints

%BEGIN Task Window, id_Display_time_based_flights
  task_win_eh(_Win,e_Menu(id_Display_time_based_flights,_ShiftCtlAlt),0):-!,
       win_time_based_flights_Create(_Win),
       !.
%END Task Window, id_Display_time_based_flights

%BEGIN Task Window, id_edit_cut
  task_win_eh(_Win,e_Menu(id_edit_cut,_ShiftCtlAlt),0):-!,
       keyword(_,Message,color_LtGray),!,
       term_bin(messages,Message,Bin),
       cb_PutBin("Message",Bin),
       delete_rectangle(_Win),
       !.
%END Task Window, id_edit_cut

%BEGIN Task Window, id_edit_copy
  task_win_eh(_Win,e_Menu(id_edit_copy,_ShiftCtlAlt),0):-!,
       keyword(_,Message,color_LtGray),!,
       term_bin(messages,Message,Bin),
       cb_PutBin("Message",Bin),
       !.
%END Task Window, id_edit_copy

%BEGIN Task Window, id_edit_paste
  task_win_eh(_Win,e_Menu(id_edit_paste,_ShiftCtlAlt),0):-!,
       paste_keyword(_Win),
       !.
%END Task Window, id_edit_paste

%BEGIN Task Window, id_edit_delete
  task_win_eh(_Win,e_Menu(id_edit_delete,_ShiftCtlAlt),0):-!,
       delete_rectangle(_Win),
       !.
%END Task Window, id_edit_delete

%BEGIN Task Window, id_edit_check_the_model
  task_win_eh(_Win,e_Menu(id_edit_check_the_model,_ShiftCtlAlt),0):-!,
       check_right_model,
       dlg_Note("Check the model","Ckeck over!"),
       !.
%END Task Window, id_edit_check_the_model

%BEGIN Task Window, e_Char
  task_win_eh(_Win,e_Char(k_del,_ShiftCtlAlt),0):-!,
       delete_rectangle(_Win),
       !.

  task_win_eh(_Win,e_Char(k_up,_ShiftCtlAlt),0):-!,
       keyword(Posit,_,color_LtGray),
       LastPosit = Posit - 1,
```

```
        move_GrayCase(_Win,LastPosit),
        winRefresh(_Win),
        !.

   task_win_eh(_Win,e_Char(k_down,_ShiftCtlAlt),0):-!,
        keyword(Posit,_,color_LtGray),
        LastPosit = Posit + 1,
        move_GrayCase(_Win,LastPosit),
        winRefresh(_Win),
        !.

   task_win_eh(_Win,e_Char(k_enter,_ShiftCtlAlt),0):-!,
        keyword(Posit,message(Text,_),color_LtGray),!,
        values_associated(_Win,Posit,Text),
        winRefresh(_Win),
        !.

   task_win_eh(_Win,e_Char(k_ctrl_x,_ShiftCtlAlt),0):-!,
        keyword(_,Message,color_LtGray),!,
        term_bin(messages,Message,Bin),
        cb_PutBin("Message",Bin),
        delete_rectangle(_Win),
        !.

   task_win_eh(_Win,e_Char(k_ctrl_c,_ShiftCtlAlt),0):-!,
        keyword(_,Message,color_LtGray),!,
        term_bin(messages,Message,Bin),
        cb_PutBin("Message",Bin),
        !.

   task_win_eh(_Win,e_Char(k_ctrl_v,_ShiftCtlAlt),0):-!,
        paste_keyword(_Win),
        !.
%END Task Window, e_Char


%BEGIN Task Window, id_help_contents
   task_win_eh(_Win,e_Menu(id_help_contents,_ShiftCtlAlt),0):-!,
        vpi_ShowHelp("script.hlp"),
        !.
%END Task Window, id_help_contents

%BEGIN Task Window, id_help_about
   task_win_eh(Win,e_Menu(id_help_about,_ShiftCtlAlt),0):-!,
        dlg_about_dialog_Create(Win),
        !.
%END Task Window, id_help_about

%BEGIN Task Window, idt_waypoint_control
   task_win_eh(_Win,e_Menu(idt_waypoint_control,_ShiftCtlAlt),0):-!,
        check_already_exist(_Win,"USE WAYPOINT CONTROL"),
        decide_toolbar(_Win),
        winRefresh(_Win),
        !.
%END Task Window, idt_waypoint_control

%BEGIN Task Window, idt_time_based_control
   task_win_eh(_Win,e_Menu(idt_time_based_control,_ShiftCtlAlt),0):-!,
        check_already_exist(_Win,"USE TIME BASED CONTROL"),
        decide_toolbar(_Win),
        winRefresh(_Win),
        !.
%END Task Window, idt_time_based_control
```

```
%BEGIN Task Window, idt_adv_power
  task_win_eh(_Win,e_Menu(idt_adv_power,_ShiftCtlAlt),0):-!,
       %toolbar_SetValue(_Win,idt_adv_power,ctrl_value(0,1)),
       build_keyword(_Win,"Turn on ADV power"),
       !.
%END Task Window, idt_adv_power

%BEGIN Task Window, idt_turn_off_adv_power
  task_win_eh(_Win,e_Menu(idt_turn_off_adv_power,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Turn off ADV power"),!.
%END Task Window, idt_turn_off_adv_power

%BEGIN Task Window, idt_sonar_power
  task_win_eh(_Win,e_Menu(idt_sonar_power,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Turn on sonar power"),
       !.
%END Task Window, idt_sonar_power

%BEGIN Task Window, idt_turn_off_sonar_power
  task_win_eh(_Win,e_Menu(idt_turn_off_sonar_power,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Turn off sonar power"),!.
%END Task Window, idt_turn_off_sonar_power

%BEGIN Task Window, idt_flight_gains
  task_win_eh(_Win,e_Menu(idt_flight_gains,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Get flight controller gains"),!.
%END Task Window, idt_flight_gains

%BEGIN Task Window, idt_motor_gains
  task_win_eh(_Win,e_Menu(idt_motor_gains,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Get motor controller gains"),!.
%END Task Window, idt_motor_gains

%BEGIN Task Window, idt_set_max_depth
  task_win_eh(_Win,e_Menu(idt_set_max_depth,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set max depth"),!.
%END Task Window, idt_set_max_depth

%BEGIN Task Window, idt_setmin_battery_voltage
  task_win_eh(_Win,e_Menu(idt_setmin_battery_voltage,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set min battery voltage"),!.
%END Task Window, idt_setmin_battery_voltage

%BEGIN Task Window, idt_initialize_boards
  task_win_eh(_Win,e_Menu(idt_initialize_boards,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Initialize boards"),!.
%END Task Window, idt_initialize_boards

%BEGIN Task Window, idt_prop_power
  task_win_eh(_Win,e_Menu(idt_prop_power,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Turn on prop power"),!.
%END Task Window, idt_prop_power

%BEGIN Task Window, idt_turn_off_prop_power
  task_win_eh(_Win,e_Menu(idt_turn_off_prop_power,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Turn off prop power"),!.
%END Task Window, idt_turn_off_prop_power

%BEGIN Task Window, idt_zero_gyros
  task_win_eh(_Win,e_Menu(idt_zero_gyros,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Zero gyros and depth cell"),!.
%END Task Window, idt_zero_gyros
```

```
%BEGIN Task Window, idt_zero_depth_cell
  task_win_eh(_Win,e_Menu(idt_zero_depth_cell,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Zero depth cell"),!.
%END Task Window, idt_zero_depth_cell

%BEGIN Task Window, idt_start_depth_filter
  task_win_eh(_Win,e_Menu(idt_start_depth_filter,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Start depth filter"),!.
%END Task Window, idt_start_depth_filter

%BEGIN Task Window, idt_ignore_leak_check
  task_win_eh(_Win,e_Menu(idt_ignore_leak_check,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Ignore leak check"),!.
%END Task Window, idt_ignore_leak_check

%BEGIN Task Window, idt_ignore_voltage_check
  task_win_eh(_Win,e_Menu(idt_ignore_voltage_check,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Ignore voltage check"),!.
%END Task Window, idt_ignore_voltage_check

%BEGIN Task Window, idt_set_gps_origin
  task_win_eh(_Win,e_Menu(idt_set_gps_origin,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set GPS origin"),!.
%END Task Window, idt_set_gps_origin

%BEGIN Task Window, idt_wait
  task_win_eh(_Win,e_Menu(idt_wait,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Wait"),!.
%END Task Window, idt_wait

%BEGIN Task Window, idt_init_done
  task_win_eh(_Win,e_Menu(idt_init_done,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Initialization done"),!.
%END Task Window, idt_init_done

%BEGIN Task Window, idt_set_screw_speed
  task_win_eh(_Win,e_Menu(idt_set_screw_speed,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set screw speed"),!.
%END Task Window, idt_set_screw_speed

%BEGIN Task Window, idt_screw_speed_from_file
  task_win_eh(_Win,e_Menu(idt_screw_speed_from_file,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set screw speed from file"),!.
%END Task Window, idt_screw_speed_from_file

%BEGIN Task Window, idt_start_screw_speed_control
  task_win_eh(_Win,e_Menu(idt_start_screw_speed_control,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Start screw speed control"),!.
%END Task Window, idt_start_screw_speed_control

%BEGIN Task Window, idt_stop_screw_speed_control
  task_win_eh(_Win,e_Menu(idt_stop_screw_speed_control,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Stop screw speed control"),!.
%END Task Window, idt_stop_screw_speed_control

%BEGIN Task Window, idt_set_waypoint
  task_win_eh(_Win,e_Menu(idt_set_waypoint,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set waypoint XY"),!.
%END Task Window, idt_set_waypoint

%BEGIN Task Window, idt_set_waypoint_gps
  task_win_eh(_Win,e_Menu(idt_set_waypoint_gps,_ShiftCtlAlt),0):-!,
```

```
      build_keyword(_Win,"Set waypoint GPS"),!.
%END Task Window, idt_set_waypoint_gps

%BEGIN Task Window, idt_set_flight_heading
  task_win_eh(_Win,e_Menu(idt_set_flight_heading,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Set flight heading"),!.
%END Task Window, idt_set_flight_heading

%BEGIN Task Window, idt_start_flight_heading_control
  task_win_eh(_Win,e_Menu(idt_start_flight_heading_control,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Start flight heading control"),!.
%END Task Window, idt_start_flight_heading_control

%BEGIN Task Window, idt_stop_flight_heading_control
  task_win_eh(_Win,e_Menu(idt_stop_flight_heading_control,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Stop flight heading control"),!.
%END Task Window, idt_stop_flight_heading_control

%BEGIN Task Window, idt_set_flight_depth
  task_win_eh(_Win,e_Menu(idt_set_flight_depth,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Set flight depth"),!.
%END Task Window, idt_set_flight_depth

%BEGIN Task Window, idt_start_flight_depth_control
  task_win_eh(_Win,e_Menu(idt_start_flight_depth_control,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Start flight depth control"),!.
%END Task Window, idt_start_flight_depth_control

%BEGIN Task Window, idt_stop_flight_depth_control
  task_win_eh(_Win,e_Menu(idt_stop_flight_depth_control,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Stop flight depth control"),!.
%END Task Window, idt_stop_flight_depth_control

%BEGIN Task Window, idt_set_flight_duration
  task_win_eh(_Win,e_Menu(idt_set_flight_duration,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Set flight duration"),!.
%END Task Window, idt_set_flight_duration

%BEGIN Task Window, idt_depth_error_filter
  task_win_eh(_Win,e_Menu(idt_depth_error_filter,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Start depth error filter"),!.
%END Task Window, idt_depth_error_filter

%BEGIN Task Window, idt_heading_error_filter
  task_win_eh(_Win,e_Menu(idt_heading_error_filter,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Start heading error filter"),!.
%END Task Window, idt_heading_error_filter

%BEGIN Task Window, idt_surge_control_on
  task_win_eh(_Win,e_Menu(idt_surge_control_on,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Surge control on"),!.
%END Task Window, idt_surge_control_on

%BEGIN Task Window, idt_surge_control_off
  task_win_eh(_Win,e_Menu(idt_surge_control_off,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Surge control off"),!.
%END Task Window, idt_surge_control_off

%BEGIN Task Window, idt_heading_and_sway
  task_win_eh(_Win,e_Menu(idt_heading_and_sway,_ShiftCtlAlt),0):-!,
      build_keyword(_Win,"Heading and sway control"),!.
%END Task Window, idt_heading_and_sway
```

```
%BEGIN Task Window, idt_submerge
  task_win_eh(_Win,e_Menu(idt_submerge,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Submerge"),!.
%END Task Window, idt_submerge

%BEGIN Task Window, idt_rotate
  task_win_eh(_Win,e_Menu(idt_rotate,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Rotate"),!.
%END Task Window, idt_rotate

%BEGIN Task Window, idt_surface
  task_win_eh(_Win,e_Menu(idt_surface,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Surface"),!.
%END Task Window, idt_surface

%BEGIN Task Window, idt_set_screw_voltage
  task_win_eh(_Win,e_Menu(idt_set_screw_voltage,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set screw voltage"),!.
%END Task Window, idt_set_screw_voltage

%BEGIN Task Window, idt_screw_voltage_control
  task_win_eh(_Win,e_Menu(idt_screw_voltage_control,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Start screw voltage control"),!.
%END Task Window, idt_screw_voltage_control

%BEGIN Task Window, idt_set_fixed_plane_angles
  task_win_eh(_Win,e_Menu(idt_set_fixed_plane_angles,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Set fixed plane angles"),!.
%END Task Window, idt_set_fixed_plane_angles

%BEGIN Task Window, idt_fixed_plane_control
  task_win_eh(_Win,e_Menu(idt_fixed_plane_control,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Start fixed plane control"),!.
%END Task Window, idt_fixed_plane_control

%BEGIN Task Window, idt_shutdown
  task_win_eh(_Win,e_Menu(idt_shutdown,_ShiftCtlAlt),0):-!,
       build_keyword(_Win,"Shutdown"),!.
%END Task Window, idt_shutdown

%BEGIN Task Window, idt_previous
  task_win_eh(_Win,e_Menu(idt_previous,_ShiftCtlAlt),0):-!,
       toolbar_remove(_win),
       tb_script_Create(_Win),
       tb_script1_Create(_Win),
       tb_project_toolbar_Create(_Win),
       decide_toolbar(_Win),
       !.
%END Task Window, idt_previous

%BEGIN Task Window, idt_next
  task_win_eh(_Win,e_Menu(idt_next,_ShiftCtlAlt),0):-!,
       toolbar_remove(_win),
       tb_script2_Create(_Win),
       tb_script1_Create(_Win),
       tb_project_toolbar_Create(_Win),
       decide_toolbar(_Win),
       !.
%END Task Window, idt_next


%BEGIN Task Window, e_MouseDown
  task_win_eh(_Win,e_MouseDown(PNT,_,mouse_button_left),0):-!,
```

```
      win_SetState(_Win,[wsf_Enabled]),
      delete_LtGray,
      rectangle_below(_Win,PNT,_),
      winRefresh(_Win),
      !.

  task_win_eh(_Win,e_MouseDown(PNT,_,mouse_button_right),0):-!,
      menu_PopUp(_Win,res_menu(id_script_popup), PNT, align_left),
      !.
%END Task Window, e_MouseDown

%BEGIN Task Window, e_MouseDbl
  task_win_eh(_Win,e_MouseDbl(PNT,_ShiftCtlAlt,mouse_button_left),0):-!,
      delete_LtGray,
      rectangle_below(_Win,PNT,Posit),
      keyword(Posit,message(Text,_),_),!,
      values_associated(_Win,Posit,Text),
      winRefresh(_Win),
      !.
%END Task Window, e_MouseDbl

%BEGIN Task Window, e_VScroll
  task_win_eh(_Win,e_VScroll(sc_ThumbTrack,Pos),0):-!,
      win_SetScrollPos(_Win,sb_Vert, Pos),
      winRefresh(_Win),
      !.

  task_win_eh(_Win,e_VScroll(sc_LineUp,_),0):-!,
      Pos = win_GetScrollPos(_Win,sb_Vert),
      NewPos = Pos-20,
      win_SetScrollPos(_Win,sb_Vert, NewPos),
      winRefresh(_Win),
      !.

  task_win_eh(_Win,e_VScroll(sc_LineDown,_),0):-!,
      Pos = win_GetScrollPos(_Win,sb_Vert),
      NewPos = Pos+20,
      win_SetScrollPos(_Win,sb_Vert, NewPos),
      winRefresh(_Win),
      !.

  task_win_eh(_Win,e_VScroll(sc_PageUp,_),0):-!,
      Pos = win_GetScrollPos(_Win,sb_Vert),
      NewPos = Pos-100,
      win_SetScrollPos(_Win,sb_Vert, NewPos),
      winRefresh(_Win),
      !.

  task_win_eh(_Win,e_VScroll(sc_PageDown,_),0):-!,
      Pos = win_GetScrollPos(_Win,sb_Vert),
      NewPos = Pos+100,
      win_SetScrollPos(_Win,sb_Vert, NewPos),
      winRefresh(_Win),
      !.
%END Task Window, e_VScroll

%BEGIN Task Window, e_Update
  task_win_eh(_Win,e_Update(_UpdateRct),0):-!,

      RCT = win_GetClientRect( _Win ),
      RCT = rct(L,T,R,B),
      Left=L+105, Top=T+40, Right=R-105,
      win_SetClip(_Win, rct(Left, Top,Right,B)),
```

```
       winRefresh(_Win),
       !.
%END Task Window, e_Update


%BEGIN Task Window, e_Size
  task_win_eh(_Win,e_Size(_Width,_Height),0):-!,

ifdef use_tbar
       toolbar_Resize(_Win),
enddef
ifdef use_message
       msg_Resize(_Win),
enddef
       win_Invalidate(_Win),
       !.
%END Task Window, e_Size

%END_WIN Task Window

/*****************************************************************************
             Invoking on-line Help
*****************************************************************************/

  project_ShowHelpContext(HelpTopic):-
       vpi_ShowHelpContext("script.hlp",HelpTopic).

/*****************************************************************************
                      Main Goal
*****************************************************************************/

goal

ifdef use_mdi
  vpi_SetAttrVal(attr_win_mdi,b_true),
enddef
ifdef ws_win
  ifdef use_3dctrl
    vpi_SetAttrVal(attr_win_3dcontrols,b_true),
  enddef
enddef
  vpi_Init(task_win_Flags,task_win_eh,task_win_Menu,"script",task_win_Title).

%BEGIN_TLB Project toolbar, 17:39:54-12.7.1999, Code automatically updated!
/*****************************************************************************
       Creation of toolbar: Project toolbar
*****************************************************************************/

clauses

  tb_project_toolbar_Create(_Parent):-
ifdef use_tbar
       toolbar_create(tb_top,0xC0C0C0,_Parent,
              [tb_ctrl(id_file_new,pushb,idb_new_up,idb_new_dn,idb_new_up,"New;New
file",1,1),
               tb_ctrl(id_file_open,pushb,idb_open_up,idb_open_dn,idb_open_up,"Open;Open
file",1,1),

tb_ctrl(id_file_save_model,pushb,idb_save_up,idb_save_dn,idb_save_up,"Save;File
save",1,1),
               separator,
               separator,
```

```
                tb_ctrl(id_edit_cut,pushb,idb_cut_up,idb_cut_dn,idb_cut_up,"Cut;Cut to
clipboard",1,1),
                tb_ctrl(id_edit_copy,pushb,idb_copy_up,idb_copy_dn,idb_copy_up,"Copy;Copy
to clipboard",1,1),

tb_ctrl(id_edit_paste,pushb,idb_paste_up,idb_paste_dn,idb_paste_up,"Paste;Paste from
clipboard",1,1),
                separator,
                separator,
                separator,
                separator,

tb_ctrl(id_edit_check_the_model,pushb,idb_check,idb_check_down,idb_check_down,"",1,1),
                separator,
                separator,
                separator,

tb_ctrl(idt_waypoint_control,pushb,idb_waypoint_control,idb_waypoint_control_down,idb_wa
ypoint_control_unable,"",0,1),

tb_ctrl(idt_time_based_control,pushb,idb_time_based_control,idb_time_based_control_down,
idb_time_based_control_unable,"",0,1)]),
enddef
        true.
%END_TLB Project toolbar




%BEGIN_TLB Help line, 16:30:51-10.5.1999, Code automatically updated!
/*****************************************************************************
        Creation of toolbar: Help line
*****************************************************************************/

clauses

  tb_help_line_Create(_Parent):-
ifdef use_tbar
        toolbar_create(tb_bottom,0xC0C0C0,_Parent,
                [tb_text(idt_help_line,tb_context,452,0,4,10,0x0,"")]),
enddef
        true.
%END_TLB Help line



%BEGIN_DLG About dialog
/*****************************************************************************
        Creation and event handling for dialog: About dialog
*****************************************************************************/

constants

%BEGIN About dialog, CreateParms, 16:30:51-10.5.1999, Code automatically updated!
  dlg_about_dialog_ResID = idd_dlg_about
  dlg_about_dialog_DlgType = wd_Modal
  dlg_about_dialog_Help = idh_contents
%END About dialog, CreateParms

predicates

  dlg_about_dialog_eh : EHANDLER
```

```
clauses

  dlg_about_dialog_Create(Parent):-
       win_CreateResDialog(Parent,dlg_about_dialog_DlgType,dlg_about_dialog_ResID,dlg_abo
ut_dialog_eh,0).

%BEGIN About dialog, idc_ok _CtlInfo
  dlg_about_dialog_eh(_Win,e_Control(idc_ok,_CtrlType,_CtrlWin,_CtrlInfo),0):-!,
       win_Destroy(_Win),
       !.
%END About dialog, idc_ok _CtlInfo
%MARK About dialog, new events

  dlg_about_dialog_eh(_,_,_):-!,fail.

%END_DLG About dialog

%BEGIN_TLB script, 14:45:15-20.7.1999, Code automatically updated!
/*****************************************************************************
       Creation of toolbar: script
*****************************************************************************/

clauses

  tb_script_Create(_Parent):-
ifdef use_tbar
       toolbar_create(tb_right,0xC0C0C0,_Parent,

       [tb_ctrl(idt_set_screw_speed,pushb,idb_set_screw_speed,idb_set_screw_speed_down,id
b_set_screw_speed_enable,"",0,1),

tb_ctrl(idt_screw_speed_from_file,pushb,idb_screw_speed_from_file,idb_screw_speed_from_f
ile_down,idb_screw_speed_from_file_unable,"",0,1),

tb_ctrl(idt_start_screw_speed_control,pushb,idb_start_screw_speed_control,idb_start_scre
w_speed_control_down,idb_start_screw_speed_control_enable,"",0,1),

tb_ctrl(idt_stop_screw_speed_control,pushb,idb_stop_screw_speed_control,idb_stop_screw_s
peed_control_down,idb_stop_screw_speed_control_enable,"",0,1),
              separator,

tb_ctrl(idt_set_waypoint,pushb,idb_set_waypoint,idb_set_waypoint_down,idb_set_waypoint_e
nable,"",0,1),

tb_ctrl(idt_set_waypoint_gps,pushb,idb_set_waypoint_gps,idb_set_waypoint_gps_down,idb_se
t_waypoint_gps_unable,"",0,1),
              separator,

tb_ctrl(idt_set_flight_heading,pushb,idb_set_flight_heading,idb_set_flight_heading_down,
idb_set_flight_heading_enable,"",0,1),

tb_ctrl(idt_start_flight_heading_control,pushb,idb_start_flight_heading_control,idb_star
t_flight_heading_control_down,idb_start_flight_heading_control_enable,"",0,1),

tb_ctrl(idt_stop_flight_heading_control,pushb,idb_stop_flight_heading_control,idb_stop_f
light_heading_control_down,idb_stop_flight_heading_control_enable,"",0,1),
              separator,

tb_ctrl(idt_set_flight_depth,pushb,idb_set_flight_depth,idb_set_flight_depth_down,idb_se
t_flight_depth_enable,"",0,1),
```

```
tb_ctrl(idt_start_flight_depth_control,pushb,idb_start_flight_depth_control,idb_start_fl
ight_depth_control_down,idb_start_flight_depth_control_enable,"",0,1),

tb_ctrl(idt_stop_flight_depth_control,pushb,idb_stop_flight_depth_control,idb_stop_fligh
t_depth_control_down,idb_stop_flight_depth_control_enable,"",0,1),
              separator,

tb_ctrl(idt_set_flight_duration,pushb,idb_set_flight_duration,idb_set_flight_duration_do
wn,idb_set_flight_duration_enable,"",0,1),
              separator,
              separator,

tb_ctrl(idt_shutdown,pushb,idb_shutdown,idb_shutdown_down,idb_shutdown_enable,"",0,1),
              separator,
              separator,
              tb_ctrl(idt_next,pushb,idb_next,idb_next_down,idb_next_down,"",1,1)]),
enddef
      true.
%END_TLB script



%BEGIN_TLB script1, 14:42:50-20.7.1999, Code automatically updated!
/*****************************************************************************
      Creation of toolbar: script1
*****************************************************************************/

clauses

  tb_script1_Create(_Parent):-
ifdef use_tbar
      toolbar_create(tb_left,0xC0C0C0,_Parent,

      [tb_ctrl(idt_adv_power,pushb,idb_adv_power,idb_adv_power_down,idb_adv_unable,"",0,
1),

tb_ctrl(idt_turn_off_adv_power,pushb,idb_turn_off_adv_power,idb_adv_power_down,idb_turn_
off_adv_unable,"",0,1),

tb_ctrl(idt_sonar_power,pushb,idb_sonar_power,idb_sonar_power_down,idb_sonar_power_unabl
e,"",0,1),

tb_ctrl(idt_turn_off_sonar_power,pushb,idb_turn_off_sonar_power,idb_turn_off_sonar_down,
idb_turn_off_sonar_unable,"",0,1),

tb_ctrl(idt_flight_gains,pushb,idb_flight_controller_gains,idb_flight_controller_gains_d
own,idb_flight_controller_gains_unable,"",0,1),

tb_ctrl(idt_motor_gains,pushb,idb_motor_controller_gains,idb_motor_controller_gains_down
,idb_motor_controller_gains_unable,"",0,1),

tb_ctrl(idt_set_max_depth,pushb,idb_set_max_depth,idb_set_max_depth_down,idb_set_max_dep
th_unable,"",0,1),

tb_ctrl(idt_setmin_battery_voltage,pushb,idb_setmin_battery_voltage,idb_setmin_battery_v
oltage_down,idb_setmin_battery_voltage_unable,"",0,1),

tb_ctrl(idt_initialize_boards,pushb,idb_initialize_boards,idb_initialize_boards_down,idb
_initialize_boards_unable,"",0,1),
```

```
tb_ctrl(idt_prop_power,pushb,idb_prop_power,idb_prop_power_down,idb_prop_power_unable,""
,0,1),

tb_ctrl(idt_turn_off_prop_power,pushb,idb_turn_off_prop_power,idb_turn_off_prop_down,idb
_turn_off_prop_enable,"",0,1),

tb_ctrl(idt_zero_gyros,pushb,idb_zero_gyros,idb_zero_gyros_down,idb_zero_gyros_unable,""
,0,1),

tb_ctrl(idt_zero_depth_cell,pushb,idb_zero_depth_cell,idb_zero_depth_cell_down,idb_zero_
depth_cell_unable,"",0,1),

tb_ctrl(idt_start_depth_filter,pushb,idb_start_depth_filter,idb_start_depth_filter_down,
idb_start_depth_filter_unable,"",0,1),

tb_ctrl(idt_ignore_leak_check,pushb,idb_ignore_leak_check,idb_ignore_leak_check_down,idb
_ignore_leak_check_unable,"",0,1),

tb_ctrl(idt_ignore_voltage_check,pushb,idb_ignore_voltage_check,idb_ignore_voltage_check
_down,idb_ignore_voltage_check_unable,"",0,1),

tb_ctrl(idt_set_gps_origin,pushb,idb_set_gps_origin,idb_set_gps_origin_down,idb_set_gps_
origin_unable,"",0,1),
              tb_ctrl(idt_wait,pushb,idb_wait,idb_wait_down,idb_wait_unable,"",0,1),

tb_ctrl(idt_init_done,pushb,idb_init_done,idb_init_done_down,idb_init_done_unable,"",0,1
)]),
enddef
      true.
%END_TLB script1



%BEGIN_TLB script2, 14:46:07-20.7.1999, Code automatically updated!
/*****************************************************************************
      Creation of toolbar: script2
*****************************************************************************/

clauses

  tb_script2_Create(_Parent):-
ifdef use_tbar
      toolbar_create(tb_right,0xC0C0C0,_Parent,

      [tb_ctrl(idt_depth_error_filter,pushb,idb_depth_error_filter,idb_depth_error_filte
r_down,idb_depth_error_filter_unable,"",0,1),

tb_ctrl(idt_heading_error_filter,pushb,idb_heading_error_filter,idb_heading_error_filter
_down,idb_heading_error_filter_unable,"",0,1),
              separator,

tb_ctrl(idt_surge_control_on,pushb,idb_surge_control_on,idb_surge_control_on_down,idb_su
rge_control_on_unable,"",0,1),

tb_ctrl(idt_surge_control_off,pushb,idb_surge_control_off,idb_surge_control_off_down,idb
_surge_control_off_unable,"",0,1),
              separator,

tb_ctrl(idt_heading_and_sway,pushb,idb_heading_and_sway,idb_heading_and_sway_down,idb_he
ading_and_sway_unable,"",0,1),
              separator,
```

```
tb_ctrl(idt_submerge,pushb,idb_submerge,idb_submerge_down,idb_submerge_unable,"",0,1),

tb_ctrl(idt_rotate,pushb,idb_rotate,idb_rotate_down,idb_rotate_unable,"",0,1),

tb_ctrl(idt_surface,pushb,idb_surface,idb_surface_down,idb_surface_unable,"",0,1),
              separator,

tb_ctrl(idt_set_screw_voltage,pushb,idb_set_screw_voltage,idb_set_screw_voltage_down,idb
_set_screw_voltage_unable,"",0,1),

tb_ctrl(idt_screw_voltage_control,pushb,idb_screw_voltage_control,idb_screw_voltage_cont
rol_down,idb_screw_voltage_control_unable,"",0,1),
              separator,

tb_ctrl(idt_set_fixed_plane_angles,pushb,idb_set_fixed_plane_angles,idb_set_fixed_plane_
angles_down,idb_set_fixed_plane_angles_unable,"",0,1),

tb_ctrl(idt_fixed_plane_control,pushb,idb_fixed_plane_control,idb_fixed_plane_control_do
wn,idb_fixed_plane_control_unable,"",0,1),
              separator,
              separator,
              separator,

tb_ctrl(idt_shutdown,pushb,idb_shutdown,idb_shutdown_down,idb_shutdown_enable,"",0,1),
              separator,
              separator,

tb_ctrl(idt_previous,pushb,idb_previous,idb_previous_down,idb_previous_down,"",1,1)]),
enddef
     true.
%END_TLB script2
```

VPITOOLS.PRO 7/29/1999

```
/*****************************************************************************

             Copyright (c) NPS

 Project:  SCRIPT
 FileName: VPITOOLS.PRO
 Purpose: Include VPI predicates and tools
 Written by: Visual Prolog Application expert
 Comments:
*****************************************************************************/

ifdef platform_16bit
  code = 5000
elsedef
% code = 48000 %set your code size > 32000 if have "Code array too small" problem
enddef

include "script.inc"
include "error.con"

/*****************************************************************************
                        Include tools
*****************************************************************************/

ifdef use_message
  include "iodecl.pre"
enddef
ifdef use_dlgpack
  include "dialog\\dialog.pro"
enddef
ifdef use_tbar
  include "toolbar\\toolbar.pro"
enddef
ifdef use_tree
  include "tree\\vpitree.pro"
enddef
ifdef use_message
  include "messages\\messages.pro"
enddef
ifdef use_socket
  include "include\\pdcsock.pro"
enddef
ifdef use_tabdlg
  include "tabdlg\\tabdlg.pro"
enddef
ifdef use_ownerdraw
  include "owndraw\\owndraw.pro"
enddef
ifdef use_dlgdir
  include "iodecl.con"
  include "dlgdir\\sort.pro"
  include "dlgdir\\dlgdir.pro"
enddef
ifdef use_grid
  include "grid\\grid.pro"
enddef
ifdef use_date
  include "date\\date.pro"
enddef
ifdef use_treebrowser
  include "treebrws\\treebrws.pro"
enddef
ifdef use_listproperty
```

```
  include "property\\property.pro"
enddef
ifdef use_palette
  include "palette\\palette.pro"
enddef
ifdef use_progress
  include "progress\\progress.pro"
enddef
ifdef use_doc
  include "html.pro"
  include "ipf.pro"
  include "rtf.pro"
  include "errhndl.pro"
enddef
```

MANAG_COMMANDS.PRO 7/29/1999

```
/*****************************************************************************

            Copyright (c) NPS

 Project:  SCRIPT
 FileName: MANAG_COMMANDS.PRO
 Purpose: Generation of a Script file
 Written by: Joel Doleac
 Comments: This program is used to manage the main menu and the buttons.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"

predicates
   start_button(WINDOW,MENU_TAG,posit,text,Integer)

Clauses

/*****************************************************************************
                        Use to manage the main menu
*****************************************************************************/


   set_enabled_menu(W) :-
       menu_Enable(W,id_file_close,b_true),
       menu_Enable(W,id_file_save_model,b_true),
       menu_Enable(W,id_file_save_as,b_true),
       menu_Enable(W,id_display_waypoints,b_true),
       menu_Enable(W,id_display_time_based_flights,b_true),
       menu_Enable(W,id_edit_cut,b_true),
       menu_Enable(W,id_edit_copy,b_true),
       menu_Enable(W,id_edit_paste,b_true),
       menu_Enable(W,id_edit_delete,b_true),
       menu_Enable(W,id_edit_check_the_model,b_true),
       menu_Enable(W,id_Script_saveScript,b_true).


/*****************************************************************************
                    Use to manage the buttons
*****************************************************************************/

   set_main_menu(_WIN,Gray):-
       toolbar_SetValue(_Win,idt_time_based_control,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_waypoint_control,ctrl_value(Gray,1)).

   set_initialization_toolbar(_Win,Gray):-
       toolbar_SetValue(_Win,idt_sonar_power,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_turn_off_sonar_power,ctrl_value(0,1)),
       toolbar_SetValue(_Win,idt_adv_power,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_turn_off_adv_power,ctrl_value(0,1)),
       toolbar_SetValue(_Win,idt_flight_gains,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_motor_gains,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_set_max_depth,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_setmin_battery_voltage,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_initialize_boards,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_prop_power,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_turn_off_prop_power,ctrl_value(0,1)),
       toolbar_SetValue(_Win,idt_zero_gyros,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_zero_depth_cell,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_start_depth_filter,ctrl_value(Gray,1)),
       toolbar_SetValue(_Win,idt_ignore_leak_check,ctrl_value(Gray,1)),
```

```
    toolbar_SetValue(_Win,idt_ignore_voltage_check,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_set_gps_origin,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_wait,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_init_done,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_shutdown,ctrl_value(Gray,1)).


set_waypoint_control_Toolbar(_Win,Gray):-
    toolbar_SetValue(_Win,idt_set_waypoint,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_set_waypoint_gps,ctrl_value(Gray,1)).

set_time_based_control_Toolbar(_Win,Gray):-
    toolbar_SetValue(_Win,idt_set_flight_heading,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_start_flight_heading_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_flight_heading_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_set_flight_depth,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_start_flight_depth_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_flight_depth_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_set_flight_duration,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_surge_control_on,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_surge_control_off,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_heading_and_sway,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_submerge,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_rotate,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_set_fixed_plane_angles,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_fixed_plane_control,ctrl_value(0,1)).

set_control_common_Toolbar(_Win,Gray):-
    toolbar_SetValue(_Win,idt_start_screw_speed_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_screw_speed_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_screw_voltage_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_set_screw_speed,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_screw_speed_from_file,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_shutdown,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_depth_error_filter,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_heading_error_filter,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_surface,ctrl_value(Gray,1)),
    toolbar_SetValue(_Win,idt_set_screw_voltage,ctrl_value(Gray,1)).



noted_toolbar(_Win,1,_):-
    toolbar_SetValue(_Win,idt_time_based_control,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_waypoint_control,ctrl_value(1,1)),
    fail.

noted_toolbar(_Win,_,"USE WAYPOINT CONTROL"):-!,
    toolbar_SetValue(_Win,idt_time_based_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_waypoint_control,ctrl_value(1,1)).

noted_toolbar(_Win,_,"USE TIME BASED CONTROL"):-!,
    toolbar_SetValue(_Win,idt_time_based_control,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_waypoint_control,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Turn on ADV power"):-!,
    toolbar_SetValue(_Win,idt_adv_power,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_turn_off_adv_power,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Turn off ADV power"):-!,
    toolbar_SetValue(_Win,idt_adv_power,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_turn_off_adv_power,ctrl_value(0,1)).
```

```
noted_toolbar(_Win,_,"Turn on sonar power"):-!,
    toolbar_SetValue(_Win,idt_sonar_power,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_turn_off_sonar_power,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Turn off sonar power"):-!,
    toolbar_SetValue(_Win,idt_sonar_power,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_turn_off_sonar_power,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Get flight controller gains"):-!,
    toolbar_SetValue(_Win,idt_flight_gains,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Get motor controller gains"):-!,
    toolbar_SetValue(_Win,idt_motor_gains,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Set max depth"):-!,
    toolbar_SetValue(_Win,idt_set_max_depth,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Set min battery voltage"):-!,
    toolbar_SetValue(_Win,idt_setmin_battery_voltage,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Initialize boards"):-!,
    toolbar_SetValue(_Win,idt_initialize_boards,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Turn on prop power"):-!,
    toolbar_SetValue(_Win,idt_prop_power,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_turn_off_prop_power,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Turn off prop power"):-!,
    toolbar_SetValue(_Win,idt_prop_power,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_turn_off_prop_power,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Zero gyros and depth cell"):-!,
    toolbar_SetValue(_Win,idt_zero_gyros,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Zero depth cell"):-!,
    toolbar_SetValue(_Win,idt_zero_depth_cell,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Start depth filter"):-!,
    toolbar_SetValue(_Win,idt_start_depth_filter,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Ignore leak check"):-!,
    toolbar_SetValue(_Win,idt_ignore_leak_check,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Ignore voltage check"):-!,
    toolbar_SetValue(_Win,idt_ignore_voltage_check,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Set GPS origin"):-!,
    toolbar_SetValue(_Win,idt_set_gps_origin,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Initialization done"):-!,
    toolbar_SetValue(_Win,idt_init_done,ctrl_value(0,1)).

noted_toolbar(_Win,Posit,"Set screw speed"):-!,
    toolbar_SetValue(_Win,idt_set_screw_voltage,ctrl_value(0,1)),
    start_button(_Win,idt_start_screw_speed_control,Posit,"Set screw speed",1).

noted_toolbar(_Win,Posit,"Set screw speed from file"):-!,
    toolbar_SetValue(_Win,idt_set_screw_voltage,ctrl_value(0,1)),
    start_button(_Win,idt_start_screw_speed_control,Posit,"Set screw speed",1).

noted_toolbar(_Win,_,"Start screw speed control"):-
    toolbar_SetValue(_Win,idt_start_screw_speed_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_screw_speed_control,ctrl_value(1,1)).
```

```
noted_toolbar(_Win,_,"Stop screw speed control"):-!,
    toolbar_SetValue(_Win,idt_start_screw_speed_control,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_stop_screw_speed_control,ctrl_value(0,1)).

noted_toolbar(_Win,Posit,"Set screw voltage"):-!,
    toolbar_SetValue(_Win,idt_set_screw_speed,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_screw_speed_from_file,ctrl_value(0,1)),
    start_button(_Win,idt_screw_voltage_control,Posit,"Set screw voltage",1).

noted_toolbar(_Win,_,"Start screw voltage control"):-!,
    toolbar_SetValue(_Win,idt_screw_voltage_control,ctrl_value(0,1)).

noted_toolbar(_Win,Posit,"Set flight heading"):-!,
    start_button(_Win,idt_start_flight_heading_control,Posit,"Set flight heading",1).

noted_toolbar(_Win,_,"Start flight heading control"):-
    toolbar_SetValue(_Win,idt_start_flight_heading_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_flight_heading_control,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Stop flight heading control"):-
    toolbar_SetValue(_Win,idt_start_flight_heading_control,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_stop_flight_heading_control,ctrl_value(0,1)).

noted_toolbar(_Win,Posit,"Set flight depth"):-!,
    start_button(_Win,idt_start_flight_depth_control,Posit,"Set flight depth",1).

noted_toolbar(_Win,_,"Start flight depth control"):-
    toolbar_SetValue(_Win,idt_start_flight_depth_control,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_stop_flight_depth_control,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Stop flight depth control"):-
    toolbar_SetValue(_Win,idt_start_flight_depth_control,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_stop_flight_depth_control,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Start depth error filter"):-
    toolbar_SetValue(_Win,idt_depth_error_filter,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Start heading error filter"):-
    toolbar_SetValue(_Win,idt_heading_error_filter,ctrl_value(0,1)).

noted_toolbar(_Win,_,"Surge control on"):-
    toolbar_SetValue(_Win,idt_surge_control_on,ctrl_value(0,1)),
    toolbar_SetValue(_Win,idt_surge_control_off,ctrl_value(1,1)).

noted_toolbar(_Win,_,"Surge control off"):-
    toolbar_SetValue(_Win,idt_surge_control_on,ctrl_value(1,1)),
    toolbar_SetValue(_Win,idt_surge_control_off,ctrl_value(0,1)).

noted_toolbar(_Win,Posit,"Set fixed plane angles"):-
    start_button(_Win,idt_fixed_plane_control,Posit,"Set fixed plane angles",1).

noted_toolbar(_Win,_,"Start fixed plane control"):-
    toolbar_SetValue(_Win,idt_fixed_plane_control,ctrl_value(0,1)).

noted_toolbar(_,_,_):-!.


start_button(_,_,Posit,Text,Loop):-
    Loop < Posit,
    keyword(Loop,message(Text,_),_),!.
start_button(Win,Menu_tag,Posit,Text,Loop):-
    Loop < Posit,!,
```

```
        NewLoop = Loop+1,
        start_button(Win,Menu_tag,Posit,Text,NewLoop).
    start_button(Win,Menu_tag,Posit,"Set screw speed",Posit):-!,
        start_button(Win,Menu_tag,Posit,"Set screw speed from file",1).
    start_button(Win,Menu_tag,Posit,_,Posit):-
        toolbar_SetValue(Win,Menu_tag,ctrl_value(1,1)).
```

```
/*****************************************************************************

            Copyright (c) NPS

 Project:  SCRIPT
 FileName: DIALOGS.PRO
 Purpose: Generation of a Script file
 Written by: Joel Doleac
 Comments: This program is called by a double click on a rectangle. It contains
           the description of all the dialogs. These dialogs are used to enter
           the values associated with the keywords.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"

%BEGIN_DLG Get flight controller gains
/*****************************************************************************
      Creation and event handling for dialog: Get flight controller gains
*****************************************************************************/

constants

%BEGIN Get flight controller gains, CreateParms, 14:19:47-13.7.1999, Code automatically
updated!
  dlg_get_flight_controller_gains_ResID = idd_get_flight_controller_gains
  dlg_get_flight_controller_gains_DlgType = wd_Modal
  dlg_get_flight_controller_gains_Help = idh_contents
%END Get flight controller gains, CreateParms

predicates

  dlg_get_flight_controller_gains_eh : EHANDLER
  %dlg_get_flight_controller_gains_handle_answer(INTEGER
EndButton,DIALOG_VAL_LIST,VALUES)
  dlg_get_flight_controller_gains_update(DIALOG_VAL_LIST,CTLID,VALUES)
  flight_controller_gains_Create(WINDOW Parent, values In, values Out)

clauses

  dlg_get_flight_controller_gains_Create(Parent,Posit):-
       keyword(Posit,message(_,Values),_),!,
       flight_controller_gains_Create(Parent,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).


flight_controller_gains_Create(Parent,[Hz,Eta_z_fl,Phi_z_fl,Eta_psi_fl,Phi_psi_fl,Z_suck
],NewValues):-
%MARK Get flight controller gains, new variables
       dialog_CreateModal(Parent,dlg_get_flight_controller_gains_ResID,"",
              [
%BEGIN Get flight controller gains, ControlList, 14:19:47-13.7.1999, Code automatically
updated!
              df(idc_hz,editreal(HZ,[range(1.0,10.0)]),nopr),
              df(idc_eta_z_fl,editreal(ETA_Z_FL,[minimum(0.0)]),nopr),
              df(idc_phi_z_fl,editreal(PHI_Z_FL,[minimum(0.0)]),nopr),
              df(idc_eta_psi_fl,editreal(ETA_PSI_FL,[minimum(0.0)]),nopr),
              df(idc_phi_psi_fl,editreal(PHI_PSI_FL,[minimum(0.0)]),nopr),
              df(idc_z_suck,editreal(Z_SUCK,[minimum(0.0)]),nopr)
%END Get flight controller gains, ControlList
              ],
```

```
            dlg_get_flight_controller_gains_eh,0,VALLIST,ANSWER),
      dlg_get_flight_controller_gains_update(VALLIST,ANSWER,NewValues).
      %dlg_get_flight_controller_gains_handle_answer(ANSWER,VALLIST,NewValues).

  /*dlg_get_flight_controller_gains_handle_answer(idc_ok,VALLIST,NewValues):-!,
      dlg_get_flight_controller_gains_update(VALLIST,NewValues).
  dlg_get_flight_controller_gains_handle_answer(idc_cancel,_,_):-!.  % Handle Esc and
Cancel here
  dlg_get_flight_controller_gains_handle_answer(_,_,_):-
      errorexit().*/


dlg_get_flight_controller_gains_update(_VALLIST,idc_ok,[_Hz,_Eta_z_fl,_Phi_z_fl,_Eta_psi
_fl,_Phi_psi_fl,_Z_suck]):-
%BEGIN Get flight controller gains, Update controls, 14:19:47-13.7.1999, Code
automatically updated!
      _HZ = dialog_VLGetreal(idc_hz,_VALLIST),
      _ETA_Z_FL = dialog_VLGetreal(idc_eta_z_fl,_VALLIST),
      _PHI_Z_FL = dialog_VLGetreal(idc_phi_z_fl,_VALLIST),
      _ETA_PSI_FL = dialog_VLGetreal(idc_eta_psi_fl,_VALLIST),
      _PHI_PSI_FL = dialog_VLGetreal(idc_phi_psi_fl,_VALLIST),
      _Z_SUCK = dialog_VLGetreal(idc_z_suck,_VALLIST),
%END Get flight controller gains, Update controls
      true.

%MARK Get flight controller gains, new events


  dlg_get_flight_controller_gains_eh(_,_,_):-!,fail.

%END_DLG Get flight controller gains




%BEGIN_DLG Get motor controller gains
/*****************************************************************************
      Creation and event handling for dialog: Get motor controller gains
*****************************************************************************/

constants

%BEGIN Get motor controller gains, CreateParms, 14:38:24-13.7.1999, Code automatically
updated!
  dlg_get_motor_controller_gains_ResID = idd_get_motor_controller_gains
  dlg_get_motor_controller_gains_DlgType = wd_Modal
  dlg_get_motor_controller_gains_Help = idh_contents
%END Get motor controller gains, CreateParms

predicates

  dlg_get_motor_controller_gains_eh : EHANDLER
  dlg_get_motor_controller_gains_update(DIALOG_VAL_LIST,CTLID,VALUES)
  motor_controller_gains_Create(WINDOW Parent, text, values In, values Out)

clauses

  dlg_get_motor_controller_gains_Create(Parent,Posit):-
      keyword(Posit,message(Text,Values),_),!,
      determin_text(1,1,Posit,Text,NewText),
      motor_controller_gains_Create(Parent,NewText,Values,NewValues),
      retract(keyword(Posit,message(Text,Values),Color)),!,
```

```
       assert(keyword(Posit,message(Text,NewValues),Color)).
       /*ETA_LS = r(0.0),
       PHI_LS = r(0.0),
       KM_RS = r(0.0),
       ETA_RS = r(0.0),
       PHI_RS = r(0.0),
       KM_RS1 = r(0.0),*/

  motor_controller_gains_Create(Parent, Text,
[ETA_LS,PHI_LS,KM_RS,ETA_RS,PHI_RS,KM_RS1], NewValues):-
%MARK Get motor controller gains, new variables
       dialog_CreateModal(Parent,dlg_get_motor_controller_gains_ResID,Text,
              [
%BEGIN Get motor controller gains, ControlList, 14:38:24-13.7.1999, Code automatically
updated!
              df(idc_eta_ls,editreal(ETA_LS,[minimum(0.0)]),nopr),
              df(idc_phi_ls,editreal(PHI_LS,[minimum(0.0)]),nopr),
              df(idc_km_rs,editreal(KM_RS,[minimum(0.0)]),nopr),
              df(idc_eta_rs,editreal(ETA_RS,[minimum(0.0)]),nopr),
              df(idc_phi_rs,editreal(PHI_RS,[minimum(0.0)]),nopr),
              df(idc_km_rs1,editreal(KM_RS1,[minimum(0.0)]),nopr)
%END Get motor controller gains, ControlList
              ],
              dlg_get_motor_controller_gains_eh,0,VALLIST,ANSWER),
       dlg_get_motor_controller_gains_update(VALLIST,ANSWER,NewValues).


dlg_get_motor_controller_gains_update(_VALLIST,idc_ok,[_ETA_LS,_PHI_LS,_KM_RS,_ETA_RS,_P
HI_RS,_KM_RS1]):-
%BEGIN Get motor controller gains, Update controls, 14:38:24-13.7.1999, Code
automatically updated!
       _ETA_LS = dialog_VLGetreal(idc_eta_ls,_VALLIST),
       _PHI_LS = dialog_VLGetreal(idc_phi_ls,_VALLIST),
       _KM_RS = dialog_VLGetreal(idc_km_rs,_VALLIST),
       _ETA_RS = dialog_VLGetreal(idc_eta_rs,_VALLIST),
       _PHI_RS = dialog_VLGetreal(idc_phi_rs,_VALLIST),
       _KM_RS1 = dialog_VLGetreal(idc_km_rs1,_VALLIST),
%END Get motor controller gains, Update controls
       true.

%MARK Get motor controller gains, new events

  dlg_get_motor_controller_gains_eh(_,_,_):-!,fail.

%END_DLG Get motor controller gains




%BEGIN_DLG Set max depth
/*************************************************************************
      Creation and event handling for dialog: Set max depth
*************************************************************************/

constants

%BEGIN Set max depth, CreateParms, 14:38:49-13.7.1999, Code automatically updated!
  dlg_set_max_depth_ResID = idd_set_max_depth
  dlg_set_max_depth_DlgType = wd_Modal
  dlg_set_max_depth_Help = idh_contents
```

```
%END Set max depth, CreateParms

predicates

  dlg_set_max_depth_eh : EHANDLER
  dlg_set_max_depth_update(DIALOG_VAL_LIST,CTLID,VALUES)
  max_depth_Create(WINDOW Parent, values In, values Out)


clauses

  dlg_set_max_depth_Create(Parent,Posit):-
       keyword(Posit,message(_,Values),_),!,
       max_depth_Create(Parent,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).
 /* dlg_set_max_depth_Create(Parent):-

       MAX_DEPTH = r(0),
       MAX_DEPTH = 0,*/

  max_depth_Create(Parent,[Max_depth],NewValues):-

%MARK Set max depth, new variables
       dialog_CreateModal(Parent,dlg_set_max_depth_ResID,"",
              [
%BEGIN Set max depth, ControlList, 14:38:49-13.7.1999, Code automatically updated!
             df(idc_max_depth,editreal(MAX_DEPTH,[range(0.0,120.0)]),nopr)
%END Set max depth, ControlList
             ],
             dlg_set_max_depth_eh,0,VALLIST,ANSWER),
       dlg_set_max_depth_update(VALLIST,ANSWER,NewValues).

/*  dlg_set_max_depth_handle_answer(idc_ok,VALLIST):-!,
       dlg_set_max_depth_update(VALLIST).
  dlg_set_max_depth_handle_answer(idc_cancel,_):-!.  % Handle Esc and Cancel here
  dlg_set_max_depth_handle_answer(_,_):-
       errorexit().*/

  dlg_set_max_depth_update(_VALLIST,idc_ok,[_MAX_DEPTH]):-
%BEGIN Set max depth, Update controls, 14:38:49-13.7.1999, Code automatically updated!
       _MAX_DEPTH = dialog_VLGetreal(idc_max_depth,_VALLIST),
%END Set max depth, Update controls
       true.

%MARK Set max depth, new events



  dlg_set_max_depth_eh(_,_,_):-!,fail.

%END_DLG Set max depth



%BEGIN_DLG Set min battery voltage
/*****************************************************************************
       Creation and event handling for dialog: Set min battery voltage
*****************************************************************************/

constants
```

```
%BEGIN Set min battery voltage, CreateParms, 14:42:19-2.6.1999, Code automatically
updated!
  dlg_set_min_battery_voltage_ResID = idd_set_min_battery_voltage
  dlg_set_min_battery_voltage_DlgType = wd_Modal
  dlg_set_min_battery_voltage_Help = idh_contents
%END Set min battery voltage, CreateParms

predicates

  dlg_set_min_battery_voltage_eh : EHANDLER
  dlg_set_min_battery_voltage_update(DIALOG_VAL_LIST,CTLID,VALUES)
  min_battery_voltage_Create(WINDOW Parent, values In, values Out)


clauses

  dlg_set_min_battery_voltage_Create(Parent,Posit):-
       keyword(Posit,message(_,Values),_),!,
       min_battery_voltage_Create(Parent,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  min_battery_voltage_Create(Parent,[VOLTAGE],NewValues):-
%MARK Set min battery voltage, new variables
       dialog_CreateModal(Parent,dlg_set_min_battery_voltage_ResID,"",
              [
%BEGIN Set min battery voltage, ControlList, 14:42:19-2.6.1999, Code automatically
updated!
              df(idc_voltage,editreal(VOLTAGE,[range(0.0,48.0)]),nopr)
%END Set min battery voltage, ControlList
              ],
              dlg_set_min_battery_voltage_eh,0,VALLIST,ANSWER),
       dlg_set_min_battery_voltage_update(VALLIST,ANSWER,NewValues).


  dlg_set_min_battery_voltage_update(_VALLIST,idc_ok,[_VOLTAGE]):-
%BEGIN Set min battery voltage, Update controls, 14:42:19-2.6.1999, Code automatically
updated!
       _VOLTAGE = dialog_VLGetreal(idc_voltage,_VALLIST),
%END Set min battery voltage, Update controls
       true.

%MARK Set min battery voltage, new events


  dlg_set_min_battery_voltage_eh(_,_,_):-!,fail.

%END_DLG Set min battery voltage



%BEGIN_DLG Wait
/*****************************************************************************
       Creation and event handling for dialog: Wait
*****************************************************************************/

constants

%BEGIN Wait, CreateParms, 14:57:04-2.6.1999, Code automatically updated!
  dlg_wait_ResID = idd_wait
  dlg_wait_DlgType = wd_Modal
  dlg_wait_Help = idh_contents
%END Wait, CreateParms
```

```
predicates

  dlg_wait_eh : EHANDLER
  dlg_wait_update(DIALOG_VAL_LIST,CTLID,VALUES)
  wait_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_wait_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       wait_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).


  wait_Create(Parent, Text, [TIME],NewValues):-
%MARK Wait, new variables
       dialog_CreateModal(Parent,dlg_wait_ResID,Text,
              [
%BEGIN Wait, ControlList, 14:57:04-2.6.1999, Code automatically updated!
              df(idc_time,editreal(TIME,[minimum(0.0)]),nopr)
%END Wait, ControlList
              ],
              dlg_wait_eh,0,VALLIST,ANSWER),
       dlg_wait_update(VALLIST,ANSWER,NewValues).



  dlg_wait_update(_VALLIST,idc_ok,[_TIME]):-
%BEGIN Wait, Update controls, 14:57:04-2.6.1999, Code automatically updated!
       _TIME = dialog_VLGetreal(idc_time,_VALLIST),
%END Wait, Update controls
       true.

%MARK Wait, new events

  dlg_wait_eh(_,_,_):-!,fail.

%END_DLG Wait



%BEGIN_DLG Set screw speed
/*****************************************************************************
       Creation and event handling for dialog: Set screw speed
*****************************************************************************/

constants

%BEGIN Set screw speed, CreateParms, 15:44:56-2.6.1999, Code automatically updated!
  dlg_set_screw_speed_ResID = idd_set_screw_speed
  dlg_set_screw_speed_DlgType = wd_Modal
  dlg_set_screw_speed_Help = idh_contents
%END Set screw speed, CreateParms

predicates

  dlg_set_screw_speed_eh : EHANDLER
  dlg_set_screw_speed_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_screw_speed_Create(WINDOW Parent, text, values In, values Out)
```

```
clauses

  dlg_set_screw_speed_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       set_screw_speed_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).


  set_screw_speed_Create(Parent,Text,[N_LS_COM,N_RS_COM],NewValues):-
%MARK Set screw speed, new variables
       dialog_CreateModal(Parent,dlg_set_screw_speed_ResID,Text,
              [
%BEGIN Set screw speed, ControlList, 15:44:56-2.6.1999, Code automatically updated!
              df(idc_n_ls_com,editreal(N_LS_COM,[range(0.0,12.0)]),nopr),
              df(idc_n_rs_com,editreal(N_RS_COM,[range(0.0,12.0)]),nopr)
%END Set screw speed, ControlList
              ],
              dlg_set_screw_speed_eh,0,VALLIST,ANSWER),
       dlg_set_screw_speed_update(VALLIST,ANSWER,NewValues).


  dlg_set_screw_speed_update(_VALLIST,idc_ok,[_N_LS_COM,_N_RS_COM]):-
%BEGIN Set screw speed, Update controls, 15:44:56-2.6.1999, Code automatically updated!
       _N_LS_COM = dialog_VLGetreal(idc_n_ls_com,_VALLIST),
       _N_RS_COM = dialog_VLGetreal(idc_n_rs_com,_VALLIST),
%END Set screw speed, Update controls
       true.

%MARK Set screw speed, new events

  dlg_set_screw_speed_eh(_,_,_):-!,fail.

%END_DLG Set screw speed



%BEGIN_DLG Set  waypoint
/*************************************************************************
      Creation and event handling for dialog: Set  waypoint
*************************************************************************/

constants

%BEGIN Set  waypoint, CreateParms, 15:06:26-20.7.1999, Code automatically updated!
  dlg_set_waypoint_ResID = idd_set_waypoint
  dlg_set_waypoint_DlgType = wd_Modal
  dlg_set_waypoint_Help = idh_contents
%END Set  waypoint, CreateParms

predicates

  dlg_set_waypoint_eh : EHANDLER
  dlg_set_waypoint_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_waypoint_Create(WINDOW Parent, text, values In, values Out)
  check_waypoints(WINDOW,posit,posit,text,values)
  check_distance(WINDOW,posit,Real,Real,Real,Real)

clauses
  check_waypoints(Parent,Posit,Prov,Text,[r(X),r(Y),_,_,_]):-
```

```
        LastPosit = Prov - 1,
        keyword(LastPosit,message(Text,[r(X1),r(Y1),_,_,_]),_),!,
        check_distance(Parent,Posit,X,Y,X1,Y1).
   check_waypoints(Parent,Posit,Prov,Text,Values):-
        Prov > 2,!,
        LastPosit = Prov - 1,
        check_waypoints(Parent,Posit,LastPosit,Text,Values).
   check_waypoints(_,_,_,_,_):-!.

   check_distance(_,_,X,Y,X1,Y1):-
        (X-X1)*(X-X1) + (Y-Y1)*(Y-Y1) >=100,!.
   check_distance(Parent,Posit,_,_,_,_):-!,
        dlg_Error("The distance between this Waypoint and the last one is less than 10
m"),
        retract(keyword(Posit,message(Text,_),Color)),!,
        assert(keyword(Posit,message(Text,[r(0.0),r(0.0),r(0.0),r(0.0),r(0.0)]),Color)),
        dlg_set_waypoint_Create(Parent,Posit).



   dlg_set_waypoint_Create(Parent,Posit):-
        keyword(Posit,message(Text,Values),_),!,
        determin_text(1,1,Posit,Text,NewText),
        set_waypoint_Create(Parent,NewText,Values,NewValues),
        check_waypoints(Parent,Posit,Posit,Text,NewValues),
        retract(keyword(Posit,message(Text,Values),Color)),!,
        assert(keyword(Posit,message(Text,NewValues),Color)).

   set_waypoint_Create(Parent,Text,[X_COM,Y_COM,Z_COM,WATCHR,TIME_OUT],NewValues):-
%MARK Set  waypoint, new variables
        dialog_CreateModal(Parent,dlg_set_waypoint_ResID,Text,
                [
%BEGIN Set  waypoint, ControlList, 15:06:26-20.7.1999, Code automatically updated!
                df(idc_x_com,editreal(X_COM,[]),nopr),
                df(idc_y_com,editreal(Y_COM,[]),nopr),
                df(idc_z_com,editreal(Z_COM,[range(0.0,300.0)]),nopr),
                df(idc_watchr,editreal(WATCHR,[minimum(0.0)]),nopr),
                df(idc_time_out,editreal(TIME_OUT,[minimum(0.0)]),nopr)
%END Set  waypoint, ControlList
                ],
                dlg_set_waypoint_eh,0,VALLIST,ANSWER),
        dlg_set_waypoint_update(VALLIST,ANSWER,NewValues).


   dlg_set_waypoint_update(_VALLIST,idc_ok,[_X_COM,_Y_COM,_Z_COM,_WATCHR,_TIME_OUT]):-
%BEGIN Set  waypoint, Update controls, 15:06:26-20.7.1999, Code automatically updated!
        _X_COM = dialog_VLGetreal(idc_x_com,_VALLIST),
        _Y_COM = dialog_VLGetreal(idc_y_com,_VALLIST),
        _Z_COM = dialog_VLGetreal(idc_z_com,_VALLIST),
        _WATCHR = dialog_VLGetreal(idc_watchr,_VALLIST),
        _TIME_OUT = dialog_VLGetreal(idc_time_out,_VALLIST),
%END Set  waypoint, Update controls
        true.

%MARK Set  waypoint, new events

   dlg_set_waypoint_eh(_,_,_):-!,fail.

%END_DLG Set  waypoint



%BEGIN_DLG Set flight heading
```

```
/*****************************************************************************
       Creation and event handling for dialog: Set flight heading
*****************************************************************************/

constants

%BEGIN Set flight heading, CreateParms, 11:07:35-12.7.1999, Code automatically updated!
  dlg_set_flight_heading_ResID = idd_set_flight_heading
  dlg_set_flight_heading_DlgType = wd_Modal
  dlg_set_flight_heading_Help = idh_contents
%END Set flight heading, CreateParms

predicates

  dlg_set_flight_heading_eh : EHANDLER
  dlg_set_flight_heading_update(DIALOG_VAL_LIST,CTLID,VALUES)
  flight_heading_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_flight_heading_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       flight_heading_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  flight_heading_Create(Parent,Text,[HEADING],NewValues):-
%MARK Set flight heading, new variables
       dialog_CreateModal(Parent,dlg_set_flight_heading_ResID,Text,
            [
%BEGIN Set flight heading, ControlList, 11:07:35-12.7.1999, Code automatically updated!
            df(idc_heading,editreal(HEADING,[range(0.0,360.0)]),nopr)
%END Set flight heading, ControlList
            ],
            dlg_set_flight_heading_eh,0,VALLIST,ANSWER),
       dlg_set_flight_heading_update(VALLIST,ANSWER,NewValues).


  dlg_set_flight_heading_update(_VALLIST,idc_ok,[_HEADING]):-
%BEGIN Set flight heading, Update controls, 11:07:35-12.7.1999, Code automatically
updated!
       _HEADING = dialog_VLGetreal(idc_heading,_VALLIST),
%END Set flight heading, Update controls
       true.

%MARK Set flight heading, new events

  dlg_set_flight_heading_eh(_,_,_):-!,fail.

%END_DLG Set flight heading



%BEGIN_DLG Set flight depth
/*****************************************************************************
       Creation and event handling for dialog: Set flight depth
*****************************************************************************/

constants

%BEGIN Set flight depth, CreateParms, 11:07:10-12.7.1999, Code automatically updated!
```

```
  dlg_set_flight_depth_ResID = idd_set_flight_depth
  dlg_set_flight_depth_DlgType = wd_Modal
  dlg_set_flight_depth_Help = idh_contents
%END Set flight depth, CreateParms

predicates

  dlg_set_flight_depth_eh : EHANDLER
  dlg_set_flight_depth_update(DIALOG_VAL_LIST,CTLID,VALUES)
  flight_depth_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_flight_depth_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       flight_depth_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  flight_depth_Create(Parent,Text,[DEPTH],NewValues):-
%MARK Set flight depth, new variables
       dialog_CreateModal(Parent,dlg_set_flight_depth_ResID,Text,
              [
%BEGIN Set flight depth, ControlList, 11:07:10-12.7.1999, Code automatically updated!
              df(idc_depth,editreal(DEPTH,[range(0.0,300.0)]),nopr)
%END Set flight depth, ControlList
              ],
              dlg_set_flight_depth_eh,0,VALLIST,ANSWER),
       dlg_set_flight_depth_update(VALLIST,ANSWER,NewValues).

  dlg_set_flight_depth_update(_VALLIST,idc_ok,[_DEPTH]):-
%BEGIN Set flight depth, Update controls, 11:07:10-12.7.1999, Code automatically
updated!
       _DEPTH = dialog_VLGetreal(idc_depth,_VALLIST),
%END Set flight depth, Update controls
       true.

%MARK Set flight depth, new events

  dlg_set_flight_depth_eh(_,_,_):-!,fail.

%END_DLG Set flight depth



%BEGIN_DLG Set flight duration
/*****************************************************************************
       Creation and event handling for dialog: Set flight duration
*****************************************************************************/

constants

%BEGIN Set flight duration, CreateParms, 11:39:11-3.6.1999, Code automatically updated!
  dlg_set_flight_duration_ResID = idd_set_flight_duration
  dlg_set_flight_duration_DlgType = wd_Modal
  dlg_set_flight_duration_Help = idh_contents
%END Set flight duration, CreateParms

predicates

  dlg_set_flight_duration_eh : EHANDLER
```

```
  dlg_set_flight_duration_update(DIALOG_VAL_LIST,CTLID,VALUES)
  flight_duration_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_flight_duration_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       flight_duration_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  flight_duration_Create(Parent,Text,[DURATION],NewValues):-
%MARK Set flight duration, new variables
       dialog_CreateModal(Parent,dlg_set_flight_duration_ResID,Text,
              [
%BEGIN Set flight duration, ControlList, 11:39:11-3.6.1999, Code automatically updated!
              df(idc_duration,editreal(DURATION,[minimum(0.0)]),nopr)
%END Set flight duration, ControlList
              ],
              dlg_set_flight_duration_eh,0,VALLIST,ANSWER),
       dlg_set_flight_duration_update(VALLIST,ANSWER,NewValues).

  dlg_set_flight_duration_update(_VALLIST,idc_ok,[_DURATION]):-
%BEGIN Set flight duration, Update controls, 11:39:11-3.6.1999, Code automatically
updated!
       _DURATION = dialog_VLGetreal(idc_duration,_VALLIST),
%END Set flight duration, Update controls
       true.

%MARK Set flight duration, new events

  dlg_set_flight_duration_eh(_,_,_):-!,fail.

%END_DLG Set flight duration




%BEGIN_DLG Heading and sway control
/*****************************************************************************
       Creation and event handling for dialog: Heading and sway control
*****************************************************************************/

constants

%BEGIN Heading and sway control, CreateParms, 12:11:03-23.6.1999, Code automatically
updated!
  dlg_heading_and_sway_control_ResID = idd_heading_and_sway_control
  dlg_heading_and_sway_control_DlgType = wd_Modal
  dlg_heading_and_sway_control_Help = idh_contents
%END Heading and sway control, CreateParms

predicates

  dlg_heading_and_sway_control_eh : EHANDLER
  dlg_heading_and_sway_control_update(DIALOG_VAL_LIST,CTLID,VALUES)
  heading_and_sway_control_Create(WINDOW Parent, text, values In, values Out)
```

```
clauses

  dlg_heading_and_sway_control_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       heading_and_sway_control_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  heading_and_sway_control_Create(Parent,Text,[Y_COM,PSI_COM],NewValues):-
%MARK Heading and sway control, new variables
       dialog_CreateModal(Parent,dlg_heading_and_sway_control_ResID,Text,
              [
%BEGIN Heading and sway control, ControlList, 12:11:03-23.6.1999, Code automatically
updated!
              df(idc_y_com,editreal(Y_COM,[]),nopr),
              df(idc_psi_com,editreal(PSI_COM,[]),nopr)
%END Heading and sway control, ControlList
              ],
              dlg_heading_and_sway_control_eh,0,VALLIST,ANSWER),
       dlg_heading_and_sway_control_update(VALLIST,ANSWER,NewValues).


    dlg_heading_and_sway_control_update(_VALLIST,idc_ok,[_Y_COM,_PSI_COM]):-
%BEGIN Heading and sway control, Update controls, 12:11:03-23.6.1999, Code automatically
updated!
       _Y_COM = dialog_VLGetreal(idc_y_com,_VALLIST),
       _PSI_COM = dialog_VLGetreal(idc_psi_com,_VALLIST),
%END Heading and sway control, Update controls
       true.

%MARK Heading and sway control, new events

  dlg_heading_and_sway_control_eh(_,_,_):-!,fail.

%END_DLG Heading and sway control


%BEGIN_DLG Submerge
/*****************************************************************************
       Creation and event handling for dialog: Submerge
*****************************************************************************/

constants

%BEGIN Submerge, CreateParms, 11:08:34-12.7.1999, Code automatically updated!
  dlg_submerge_ResID = idd_submerge
  dlg_submerge_DlgType = wd_Modal
  dlg_submerge_Help = idh_contents
%END Submerge, CreateParms

predicates

  dlg_submerge_eh : EHANDLER
  dlg_submerge_update(DIALOG_VAL_LIST,CTLID,VALUES)
  submerge_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_submerge_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
```

```
       submerge_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).

  submerge_Create(Parent,Text,[Z_COM,T_Z_F,SUBMERGE_MODE],NewValues):-
%MARK Submerge, new variables

       dialog_CreateModal(Parent,dlg_submerge_ResID,Text,
            [
%BEGIN Submerge, ControlList, 11:08:34-12.7.1999, Code automatically updated!
            df(idc_z_com,editreal(Z_COM,[range(0.0,300.0)]),nopr),
            df(idc_t_z_f,editreal(T_Z_F,[]),nopr),
            df(idc_sub_mode,editreal(SUBMERGE_MODE,[minimum(0.0)]),nopr)
%END Submerge, ControlList
            ],
            dlg_submerge_eh,0,VALLIST,ANSWER),
       dlg_submerge_update(VALLIST,ANSWER,NewValues).


  dlg_submerge_update(_VALLIST,idc_ok,[_Z_COM,_T_Z_F,_SUBMERGE_MODE]):-
%BEGIN Submerge, Update controls, 11:08:34-12.7.1999, Code automatically updated!
       _Z_COM = dialog_VLGetreal(idc_z_com,_VALLIST),
       _T_Z_F = dialog_VLGetreal(idc_t_z_f,_VALLIST),
       _SUBMERGE_MODE = dialog_VLGetreal(idc_sub_mode,_VALLIST),
%END Submerge, Update controls
       true.

%MARK Submerge, new events

  dlg_submerge_eh(_,_,_):-!,fail.

%END_DLG Submerge



%BEGIN_DLG Rotate
/*****************************************************************************
       Creation and event handling for dialog: Rotate
*****************************************************************************/

constants

%BEGIN Rotate, CreateParms, 12:13:23-23.6.1999, Code automatically updated!
  dlg_rotate_ResID = idd_rotate
  dlg_rotate_DlgType = wd_Modal
  dlg_rotate_Help = idh_contents
%END Rotate, CreateParms

predicates

  dlg_rotate_eh : EHANDLER
  dlg_rotate_update(DIALOG_VAL_LIST,CTLID,VALUES)
  rotate_Create(WINDOW Parent, text, values In, values Out)

clauses

  dlg_rotate_Create(Parent,Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       rotate_Create(Parent,NewText,Values,NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).
```

```
  rotate_Create(Parent,Text,[PSI_COM,T_PSI_F,ROTATE_MODE],NewValues):-
%MARK Rotate, new variables
      dialog_CreateModal(Parent,dlg_rotate_ResID,Text,
            [
%BEGIN Rotate, ControlList, 12:13:23-23.6.1999, Code automatically updated!
            df(idc_psi_com,editreal(PSI_COM,[]),nopr),
            df(idc_t_psi_f,editreal(T_PSI_F,[]),nopr),
            df(idc_rotate_mode,editreal(ROTATE_MODE,[minimum(0.0)]),nopr)
%END Rotate, ControlList
            ],
            dlg_rotate_eh,0,VALLIST,ANSWER),
      dlg_rotate_update(VALLIST,ANSWER,NewValues).


  dlg_rotate_update(_VALLIST,idc_ok,[_PSI_COM,_T_PSI_F,_ROTATE_MODE]):-
%BEGIN Rotate, Update controls, 12:13:23-23.6.1999, Code automatically updated!
      _PSI_COM = dialog_VLGetreal(idc_psi_com,_VALLIST),
      _T_PSI_F = dialog_VLGetreal(idc_t_psi_f,_VALLIST),
      _ROTATE_MODE = dialog_VLGetreal(idc_rotate_mode,_VALLIST),
%END Rotate, Update controls
      true.

%MARK Rotate, new events

  dlg_rotate_eh(_,_,_):-!,fail.

%END_DLG Rotate


%BEGIN_DLG Set screw voltage
/*****************************************************************************
      Creation and event handling for dialog: Set screw voltage
*****************************************************************************/

constants

%BEGIN Set screw voltage, CreateParms, 12:14:43-23.6.1999, Code automatically updated!
  dlg_set_screw_voltage_ResID = idd_set_screw_voltage
  dlg_set_screw_voltage_DlgType = wd_Modal
  dlg_set_screw_voltage_Help = idh_contents
%END Set screw voltage, CreateParms

predicates

  dlg_set_screw_voltage_eh : EHANDLER
  dlg_set_screw_voltage_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_screw_voltage_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_screw_voltage_Create(Parent,Posit):-
      keyword(Posit,message(Text,Values),_),!,
      determin_text(1,1,Posit,Text,NewText),
      set_screw_voltage_Create(Parent,NewText,Values,NewValues),
      retract(keyword(Posit,message(Text,Values),Color)),!,
      assert(keyword(Posit,message(Text,NewValues),Color)).

  set_screw_voltage_Create(Parent,Text,[SCREW_VOLTAGE],NewValues):-
%MARK Set screw voltage, new variables

      dialog_CreateModal(Parent,dlg_set_screw_voltage_ResID,Text,
```

```
            [
%BEGIN Set screw voltage, ControlList, 12:14:43-23.6.1999, Code automatically updated!
            df(idc_screw_voltage,editreal(SCREW_VOLTAGE,[range(-24.0,24.0)]),nopr)
%END Set screw voltage, ControlList
            ],
            dlg_set_screw_voltage_eh,0,VALLIST,ANSWER),
      dlg_set_screw_voltage_update(VALLIST,ANSWER,NewValues).


  dlg_set_screw_voltage_update(_VALLIST,idc_ok,[_SCREW_VOLTAGE]):-
%BEGIN Set screw voltage, Update controls, 12:14:43-23.6.1999, Code automatically
updated!
      _SCREW_VOLTAGE = dialog_VLGetreal(idc_screw_voltage,_VALLIST),
%END Set screw voltage, Update controls
      true.

%MARK Set screw voltage, new events

  dlg_set_screw_voltage_eh(_,_,_):-!,fail.

%END_DLG Set screw voltage


%BEGIN_DLG Set fixed plane angles
/*************************************************************************
      Creation and event handling for dialog: Set fixed plane angles
*************************************************************************/

constants

%BEGIN Set fixed plane angles, CreateParms, 11:09:50-12.7.1999, Code automatically
updated!
  dlg_set_fixed_plane_angles_ResID = idd_set_fixed_plane_angles
  dlg_set_fixed_plane_angles_DlgType = wd_Modal
  dlg_set_fixed_plane_angles_Help = idh_contents
%END Set fixed plane angles, CreateParms

predicates

  dlg_set_fixed_plane_angles_eh : EHANDLER
  dlg_set_fixed_plane_angles_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_fixed_plane_angles_Create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_fixed_plane_angles_Create(Parent,Posit):-
      keyword(Posit,message(Text,Values),_),!,
      determin_text(1,1,Posit,Text,NewText),
      set_fixed_plane_angles_Create(Parent,NewText,Values,NewValues),
      retract(keyword(Posit,message(Text,Values),Color)),!,
      assert(keyword(Posit,message(Text,NewValues),Color)).

  set_fixed_plane_angles_Create(Parent,Text,[PLANEANGLE,RUDDERANGLE],NewValues):-
%MARK Set fixed plane angles, new variables

      dialog_CreateModal(Parent,dlg_set_fixed_plane_angles_ResID,Text,
            [
%BEGIN Set fixed plane angles, ControlList, 11:09:50-12.7.1999, Code automatically
updated!
            df(idc_planeangle,editreal(PLANEANGLE,[range(0.0,360.0)]),nopr),
            df(idc_rudderangle,editreal(RUDDERANGLE,[range(0.0,360.0)]),nopr)
%END Set fixed plane angles, ControlList
```

```
            ],
            dlg_set_fixed_plane_angles_eh,0,VALLIST,ANSWER),
      dlg_set_fixed_plane_angles_update(VALLIST,ANSWER,NewValues).


  dlg_set_fixed_plane_angles_update(_VALLIST,idc_ok,[_PLANEANGLE,_RUDDERANGLE]):-
%BEGIN Set fixed plane angles, Update controls, 11:09:50-12.7.1999, Code automatically
updated!
      _PLANEANGLE = dialog_VLGetreal(idc_planeangle,_VALLIST),
      _RUDDERANGLE = dialog_VLGetreal(idc_rudderangle,_VALLIST),
%END Set fixed plane angles, Update controls
      true.

%MARK Set fixed plane angles, new events

  dlg_set_fixed_plane_angles_eh(_,_,_):-!,fail.

%END_DLG Set fixed plane angles



%BEGIN_DLG Set waypoint GPS
/*************************************************************************
      Creation and event handling for dialog: Set waypoint GPS
*************************************************************************/

constants

%BEGIN Set waypoint GPS, CreateParms, 17:07:16-20.7.1999, Code automatically updated!
  dlg_set_waypoint_gps_ResID = idd_set_waypoint_gps
  dlg_set_waypoint_gps_DlgType = wd_Modal
  dlg_set_waypoint_gps_Help = idh_contents
%END Set waypoint GPS, CreateParms

predicates

  dlg_set_waypoint_gps_eh : EHANDLER
  dlg_set_waypoint_gps_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_waypoint_gps_create(WINDOW Parent, text, values In, values Out)
  check_sign(Integer, Integer)

clauses

  dlg_set_waypoint_gps_Create(Parent, Posit):-
      keyword(Posit,message(Text,Values),_),!,
      determin_text(1,1,Posit,Text,NewText),
      set_waypoint_gps_create(Parent, NewText, Values, NewValues),
      retract(keyword(Posit,message(Text,Values),Color)),!,
      assert(keyword(Posit,message(Text,NewValues),Color)).


  set_waypoint_gps_create(Parent, Text, [r(LongR),r(LatR),Z_COM,WATCHR,TIME_OUT],
NewValues):-
      Long = val(integer,LongR), Lat = val(integer,LatR),
      LoDeg = Long div (3600*1000), Rest1 = abs(Long) mod (3600*1000),
      LONG_DEG_ = i(LoDeg),
      LoMin = Rest1 div (60*1000), Rest2 = Rest1 mod (60*1000),
      LONG_MIN = i(LoMin),
      LoSec = Rest2 div 1000, LoMSec = Rest2 mod 1000,
      LONG_SEC = i(LoSec), LONG_MSEC = i(LoMSec),
      LaDeg = Lat div (3600*1000), Reste1 = abs(Lat) mod (3600*1000),
      LAT_DEG = i(LaDeg),
      LaMin = Reste1 div (60*1000), Reste2 = Reste1 mod (60*1000),
```

```
        LAT_MIN = i(LaMin),
        LaSec = Reste2 div 1000, LaMSec = Reste2 mod 1000,
        LAT_SEC = i(LaSec), LAT_MSEC = i(LaMSec),
%MARK Set waypoint GPS, new variables
        dialog_CreateModal(Parent,dlg_set_waypoint_gps_ResID,Text,
                [
%BEGIN Set waypoint GPS, ControlList, 17:07:16-20.7.1999, Code automatically updated!
                df(idc_long_deg,editint(LONG_DEG_,[range(-180,180)]),nopr),
                df(idc_long_min,editint(LONG_MIN,[range(0,59)]),nopr),
                df(idc_long_sec,editint(LONG_SEC,[range(0,59)]),nopr),
                df(idc_long_msec,editint(LONG_MSEC,[range(0,999)]),nopr),
                df(idc_lat_deg,editint(LAT_DEG,[range(-90,90)]),nopr),
                df(idc_lat_min,editint(LAT_MIN,[range(0,59)]),nopr),
                df(idc_lat_sec,editint(LAT_SEC,[range(0,59)]),nopr),
                df(idc_lat_msec,editint(LAT_MSEC,[range(0,999)]),nopr),
                df(idc_z_com,editreal(Z_COM,[range(0.0,120.0)]),nopr),
                df(idc_watchr,editreal(WATCHR,[minimum(0.0)]),nopr),
                df(idc_time_out,editreal(TIME_OUT,[minimum(0.0)]),nopr)
%END Set waypoint GPS, ControlList
                ],
                dlg_set_waypoint_gps_eh,0,VALLIST,ANSWER),
        dlg_set_waypoint_gps_update(VALLIST,ANSWER,NewValues).


dlg_set_waypoint_gps_update(_VALLIST,idc_ok,[r(_LongR),r(_LatR),_Z_COM,_WATCHR,_TIME_OUT
]):-
%BEGIN Set waypoint GPS, Update controls, 17:07:16-20.7.1999, Code automatically
updated!
        _LONG_DEG_ = dialog_VLGetint(idc_long_deg,_VALLIST),
        _LONG_MIN = dialog_VLGetint(idc_long_min,_VALLIST),
        _LONG_SEC = dialog_VLGetint(idc_long_sec,_VALLIST),
        _LONG_MSEC = dialog_VLGetint(idc_long_msec,_VALLIST),
        _LAT_DEG = dialog_VLGetint(idc_lat_deg,_VALLIST),
        _LAT_MIN = dialog_VLGetint(idc_lat_min,_VALLIST),
        _LAT_SEC = dialog_VLGetint(idc_lat_sec,_VALLIST),
        _LAT_MSEC = dialog_VLGetint(idc_lat_msec,_VALLIST),
        _Z_COM = dialog_VLGetreal(idc_z_com,_VALLIST),
        _WATCHR = dialog_VLGetreal(idc_watchr,_VALLIST),
        _TIME_OUT = dialog_VLGetreal(idc_time_out,_VALLIST),
%END Set waypoint GPS, Update controls
        _LONG_DEG_ = i(LoDeg), check_sign(LoDeg, Sign),
        _LONG_MIN = i(LoMin),_LONG_SEC = i(LoSec),_LONG_MSEC = i(LoMsec),
        _Long = Sign*((abs(LoDeg)*3600 + LoMin*60 + LoSec)*1000 + LoMsec),
        _LAT_DEG = i(LaDeg), check_sign(LaDeg, Sign2),
        _LAT_MIN = i(LaMin), _LAT_SEC = i(LaSec), _LAT_MSEC = i(LaMSec),
        _Lat = Sign2*((abs(LaDeg)*3600 + LaMin*60 + LaSec)*1000 + LaMSec),
        _LongR = val(real,_Long),
        _LatR = val(real,_Lat),
        true.

check_sign(Coord, -1):-
        Coord < 0,!.
check_sign(_, 1):-!.

%MARK Set waypoint GPS, new events

  dlg_set_waypoint_gps_eh(_,_,_):-!,fail.

%END_DLG Set waypoint GPS
```

```
%BEGIN_DLG Set GPS origin
/*****************************************************************************
      Creation and event handling for dialog: Set GPS origin
*****************************************************************************/

constants

%BEGIN Set GPS origin, CreateParms, 10:16:28-26.7.1999, Code automatically updated!
  dlg_set_gps_origin_ResID = idd_set_gps_origin
  dlg_set_gps_origin_DlgType = wd_Modal
  dlg_set_gps_origin_Help = idh_contents
%END Set GPS origin, CreateParms

predicates

  dlg_set_gps_origin_eh : EHANDLER
  dlg_set_gps_origin_update(DIALOG_VAL_LIST,CTLID,VALUES)
  set_gps_origin_create(WINDOW Parent, text, values In, values Out)


clauses

  dlg_set_gps_origin_Create(Parent, Posit):-
       keyword(Posit,message(Text,Values),_),!,
       determin_text(1,1,Posit,Text,NewText),
       set_gps_origin_create(Parent, NewText, Values, NewValues),
       retract(keyword(Posit,message(Text,Values),Color)),!,
       assert(keyword(Posit,message(Text,NewValues),Color)).


  set_gps_origin_create(Parent, Text, [r(LongR),r(LatR)], NewValues):-
       Long = val(integer,LongR),
       Lat = val(integer,LatR),
       LoDeg = Long div (3600*1000), Rest1 = abs(Long) mod (3600*1000),
       LONG_DEG = i(LoDeg),
       LoMin = Rest1 div (60*1000), Rest2 = Rest1 mod (60*1000),
       LONG_MIN = i(LoMin),
       LoSec = Rest2 div 1000, LoMSec = Rest2 mod 1000,
       LONG_SEC = i(LoSec), LONG_MSEC = i(LoMSec),
       LaDeg = Lat div (3600*1000), Reste1 = abs(Lat) mod (3600*1000),
       LAT_DEG = i(LaDeg),
       LaMin = Reste1 div (60*1000), Reste2 = Reste1 mod (60*1000),
       LAT_MIN = i(LaMin),
       LaSec = Reste2 div 1000, LaMSec = Reste2 mod 1000,
       LAT_SEC = i(LaSec), LAT_MSEC = i(LaMSec),
%MARK Set GPS origin, new variables

       dialog_CreateModal(Parent,dlg_set_gps_origin_ResID,Text,
              [
%BEGIN Set GPS origin, ControlList, 10:16:28-26.7.1999, Code automatically updated!
              df(idc_long_deg,editint(LONG_DEG,[range(-180,180)]),nopr),
              df(idc_long_min,editint(LONG_MIN,[range(0,59)]),nopr),
              df(idc_long_sec,editint(LONG_SEC,[range(0,59)]),nopr),
              df(idc_long_msec,editint(LONG_MSEC,[range(0,999)]),nopr),
              df(idc_lat_deg,editint(LAT_DEG,[range(-90,90)]),nopr),
              df(idc_lat_min,editint(LAT_MIN,[range(0,59)]),nopr),
              df(idc_lat_sec,editint(LAT_SEC,[range(0,59)]),nopr),
              df(idc_lat_msec,editint(LAT_MSEC,[range(0,999)]),nopr)
%END Set GPS origin, ControlList
              ],
              dlg_set_gps_origin_eh,0,VALLIST,ANSWER),
       dlg_set_gps_origin_update(VALLIST,ANSWER,NewValues).
```

```
  dlg_set_gps_origin_update(_VALLIST,idc_ok,[r(_LongR),r(_LatR)]):-
%BEGIN Set GPS origin, Update controls, 10:16:28-26.7.1999, Code automatically updated!
       _LONG_DEG = dialog_VLGetint(idc_long_deg,_VALLIST),
       _LONG_MIN = dialog_VLGetint(idc_long_min,_VALLIST),
       _LONG_SEC = dialog_VLGetint(idc_long_sec,_VALLIST),
       _LONG_MSEC = dialog_VLGetint(idc_long_msec,_VALLIST),
       _LAT_DEG = dialog_VLGetint(idc_lat_deg,_VALLIST),
       _LAT_MIN = dialog_VLGetint(idc_lat_min,_VALLIST),
       _LAT_SEC = dialog_VLGetint(idc_lat_sec,_VALLIST),
       _LAT_MSEC = dialog_VLGetint(idc_lat_msec,_VALLIST),
%END Set GPS origin, Update controls
       _LONG_DEG = i(LoDeg), check_sign(LoDeg, Sign),
       _LONG_MIN = i(LoMin),_LONG_SEC = i(LoSec),_LONG_MSEC = i(LoMsec),
       _Long = Sign*((abs(LoDeg)*3600 + LoMin*60 + LoSec)*1000 + LoMsec),
       _LAT_DEG = i(LaDeg), check_sign(LaDeg, Sign2),
       _LAT_MIN = i(LaMin), _LAT_SEC = i(LaSec), _LAT_MSEC = i(LaMSec),
       _Lat = Sign2*((abs(LaDeg)*3600 + LaMin*60 + LaSec)*1000 + LaMSec),
       _LongR = val(real,_Long),
       _LatR = val(real,_Lat),
       true.


%MARK Set GPS origin, new events

  dlg_set_gps_origin_eh(_,_,_):-!,fail.

%END_DLG Set GPS origin
```

```
/*****************************************************************************

             Copyright (c) NPS

 Project:  SCRIPT
 FileName: WAYPOINTS.PRO
 Purpose: Generation of a Script file
 Written by: Joel Doleac
 Comments: This program is used to create the window where is diplayed the
           commanded trajectory for waypoint control.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"

%BEGIN_WIN Waypoints
/*****************************************************************************
        Creation and event handling for window: Waypoints
*****************************************************************************/

constants
%BEGIN Waypoints, CreateParms, 11:56:52-26.7.1999, Code automatically updated!
  win_waypoints_WinType = w_TopLevel
  win_waypoints_Flags =
[wsf_SizeBorder,wsf_TitleBar,wsf_Maximize,wsf_Close,wsf_ClipSiblings]
  win_waypoints_RCT = rct(100,80,600,580)
  win_waypoints_Menu = no_menu
  win_waypoints_Title = "Waypoints XY"
  win_waypoints_Help = idh_contents
%END Waypoints, CreateParms

database - scales
  determ limit_coordonates(integer,integer,integer,integer)

predicates

  draw_route(WINDOW)
  find_order(WINDOW,Integer,posit,posit)
  decide_drawing(WINDOW,Integer,posit,posit)
  convert(WINDOW,integer,integer,integer,integer)
  convert1(WINDOW,integer,integer,integer,integer)
  find_limits(posit)
  compare_limits(integer,integer,integer,integer,integer,integer,integer)
  draw_axis(WINDOW)
  scale_axis(Integer, Integer)
  axis_lines(WINDOW, Integer)

clauses
  draw_route(Win):-
       draw_axis(Win),
       find_order(Win,1,1,1).

  find_order(Win,Nb,PrecWP,Posit):-
       counter(Num),
       Posit < Num+1,
       keyword(Posit,message("Set waypoint XY",_),_),!,
       Pen = pen(1 , ps_Solid, color_Black),
       win_SetPen(Win, Pen),
       win_SetForeColor(Win, color_Black),
       decide_drawing(Win,Nb,PrecWP,Posit),
       NewPosit = Posit+1,
       NewNb = Nb+1,
```

```
    find_order(Win,NewNb,Posit,NewPosit).
find_order(_,_,_,Posit):-
    counter(Num),
    Posit > Num,!.

find_order(Win,Nb,PrecWP,Posit):-!,
    NewPosit = Posit+1,
    find_order(Win,Nb,PrecWP,NewPosit).


decide_drawing(Win,Nb,1,WP):-!,
    keyword(WP,message("Set waypoint XY",[r(X2),r(Y2),r(Z2),r(R2),_]),_),!,
    X2p = X2*100, Y2p = Y2*100,
    X2int = val(integer,X2p),
    Y2int = val(integer,Y2p),
    convert(Win,X2int,Y2int,XW2,YW2),
    Xt = XW2-5, Yt = YW2-50,
    format(Text,"% (%, %, %)",Nb,X2,Y2,Z2),
    draw_Text(Win, Yt, Xt, Text),
    convert(Win,0,0,XW1,YW1),
    Xt0 = XW1-5, Yt0 = YW1-50,
    draw_Text(Win, Yt0, Xt0, "O (0, 0, 0)"),
    draw_Line(Win,pnt(YW1,XW1),pnt(YW2,XW2)),
    Brp = (X2+R2)*100, Trp = (X2-R2)*100,
    Lrp = (Y2-R2)*100, Rrp = (Y2+R2)*100,
    Bint = val(integer,Brp),
    Lint = val(integer,Lrp),
    Tint = val(integer,Trp),
    Rint = val(integer,Rrp),
    convert(Win,Bint,Lint,B,L),
    convert(Win,Tint,Rint,T,R),
    Pen = pen(1 , ps_Solid, color_Red),
    win_SetPen(Win, Pen),
    draw_Arc (Win, rct(L,T,R,B), L, XW2, L, XW2).


decide_drawing(Win,Nb,PrecWP,WP):-
    keyword(PrecWP,message("Set waypoint XY",[r(X1),r(Y1),_,_,_]),_),!,
    X1p = X1*100, Y1p = Y1*100,
    X1int = val(integer,X1p),
    Y1int = val(integer,Y1p),
    convert(Win,X1int,Y1int,XW1,YW1),
    keyword(WP,message("Set waypoint XY",[r(X2),r(Y2),r(Z2),r(R2),_]),_),!,
    X2p = X2*100, Y2p = Y2*100,
    X2int = val(integer,X2p),
    Y2int = val(integer,Y2p),
    convert(Win,X2int,Y2int,XW2,YW2),
    Xp = XW2-5, Yp = YW2-50,
    format(Text,"% (%, %, %)",Nb,X2,Y2,Z2),
    draw_Text(Win, Yp, Xp, Text),
    draw_Line(Win,pnt(YW1,XW1),pnt(YW2,XW2)),
    Brp = (X2+R2)*100, Trp = (X2-R2)*100,
    Lrp = (Y2-R2)*100, Rrp = (Y2+R2)*100,
    Bint = val(integer,Brp),
    Lint = val(integer,Lrp),
    Tint = val(integer,Trp),
    Rint = val(integer,Rrp),
    convert(Win,Bint,Lint,B,L),
    convert(Win,Tint,Rint,T,R),
    Pen = pen(1 , ps_Solid, color_Red),
    win_SetPen(Win, Pen),
    draw_Arc (Win, rct(L,T,R,B), L, XW2, L, XW2).
```

```
  convert(Win,Xint,Yint,XW,YW):-
       limit_coordonates(XMin,XMax,YMin,YMax),
       RCT = win_GetClientRect( Win ),
       RCT = rct(L,T,R,B),
       ((R-L)*(XMax-XMin)) <= ((YMax-YMin)*(B-T)),!,
       XW = ((XMax - 2*Xint + XMin)*(R - L) + (YMax - YMin)*(B - T))div (2*(YMax -
YMin)),
       YW = (Yint - YMin)*(R-L) div (YMax-YMin).
  convert(Win,Xint,Yint,XW,YW):-
       limit_coordonates(XMin,XMax,YMin,YMax),
       RCT = win_GetClientRect( Win ),
       RCT = rct(L,T,R,B),
       ((R-L)*(XMax-XMin)) >= ((YMax-YMin)*(B-T)),!,
       YW = ((2*Yint - YMax - YMin)*(B - T) div (XMax - XMin) + R - L) div 2,
       XW = (XMax - Xint)*(B - T) div (XMax - XMin).

  draw_axis(Win):-
       limit_coordonates(XMin,XMax,YMin,YMax),
       XMax-XMin < YMax-YMin,!,
       XDiff = XMax - XMin,
       scale_axis(XDiff,Scale),
       axis_lines(Win,Scale).

  draw_axis(Win):-!,
       limit_coordonates(_,_,YMin,YMax),
       Ydiff = YMax-YMin,
       scale_axis(YDiff,Scale),
       axis_lines(Win,Scale).


  axis_lines(Win,Scale):-
       limit_coordonates(XMin,XMax,YMin,YMax),
       X1 = Xmin + (XMax-XMin)div 10,
       X2 = X1 + Scale,
       Y1 = YMin + (Ymax-YMin)div 10,
       Y2 = Y1 + Scale,
       convert(Win,X1,Y1,XW1,YW1),
       convert(Win,X2,Y2,XW2,YW2),
       Pen = pen(1 , ps_Solid, color_Cyan),
       win_SetPen(Win, Pen),
       draw_Line(Win,pnt(YW1,XW1),pnt(YW2,XW1)),
       draw_Line(Win,pnt(YW1,XW1),pnt(YW1,XW2)),
       Xhf = X2 - Scale div 10,
       Yhfl = Y1 - Scale div 10,
       Yhfr = Y1 + Scale div 10,
       convert(Win,Xhf,Yhfl,XWhf,YWhfl),
       convert(Win,0,Yhfr,_,YWhfr),
       draw_Line(Win,pnt(YWhfl,XWhf),pnt(YW1,XW2)),
       draw_Line(Win,pnt(YWhfr,XWhf),pnt(YW1,XW2)),
       Yrf = Y2 - Scale div 10,
       Xrft = X1 + Scale div 10,
       Xrfb = X1 - Scale div 10,
       convert(Win,Xrft,Yrf,XWrft,YWrf),
       convert(Win,Xrfb,0,XWrfb,_),
       draw_Line(Win,pnt(YWrf,XWrft),pnt(YW2,XW1)),
       draw_Line(Win,pnt(YWrf,XWrfb),pnt(YW2,XW1)),
       win_SetForeColor(Win, color_Cyan),
       draw_Text(Win, YW1, XW2, "N (X)"),
       draw_Text(Win, YW2, XW1, "E (Y)").
```

```
scale_axis(Diff,20):-
     Diff<=1000,!.
scale_axis(Diff,200):-
     Diff<=10000,!.
scale_axis(Diff,2000):-
     Diff<=100000,!.
scale_axis(_,20000):-!.



find_limits(Posit):-
     keyword(Posit,message("Set waypoint XY",[r(X),r(Y),_,_,_]),_),!,
     Xp = X*100, Yp = Y*100,
     Xint = val(integer,Xp),
     Yint = val(integer,Yp),
     retract(limit_coordonates(XMin,XMax,YMin,YMax)),
     compare_limits(Xint,Yint,XMin,XMax,YMin,YMax,1),
     NewPosit=Posit+1,
     find_limits(NewPosit).
find_limits(Posit):-
     counter(Num),
     Posit >= Num+1,!,
     retract(limit_coordonates(XMin,XMax,YMin,YMax)),
     MinX = XMin - 1 -(XMax-XMin)*25 div 100,
     MaxX = XMax + 1 + (XMax-XMin)*25 div 100,
     MinY = YMin - 1 -(YMax-YMin)*25 div 100,
     MaxY = YMax + 1 + (YMax-YMin)*25 div 100,
     assert(limit_coordonates(MinX,MaxX,MinY,MaxY)).
find_limits(Posit):-!,
     NewPosit=Posit+1,
     find_limits(NewPosit).

compare_limits(X,Y,XMin,XMax,YMin,YMax,1):-
     X<XMin,!,
     compare_limits(X,Y,X,XMax,YMin,YMax,2).
compare_limits(X,Y,XMin,XMax,YMin,YMax,2):-
     X>XMax,!,
     compare_limits(X,Y,XMin,X,YMin,YMax,3).
compare_limits(X,Y,XMin,XMax,YMin,YMax,3):-
     Y<YMin,!,
     compare_limits(X,Y,XMin,XMax,Y,YMax,4).
compare_limits(_,Y,XMin,XMax,YMin,YMax,4):-
     Y>YMax,!,
     assert(limit_coordonates(XMin,XMax,YMin,Y)).
compare_limits(_,_,XMin,XMax,YMin,YMax,4):-!,
     assert(limit_coordonates(XMin,XMax,YMin,YMax)).
compare_limits(X,Y,XMin,XMax,YMin,YMax,Comp):-!,
     NextComp = Comp + 1,
     compare_limits(X,Y,XMin,XMax,YMin,YMax,NextComp).

convert1(Win,Xint,Yint,XW,YW):-
     limit_coordonates(XMin,XMax,YMin,YMax),
     RCT = win_GetClientRect( Win ),
     RCT = rct(L,T,R,B),
     ((R-L)*(XMax-XMin)) <= ((YMax-YMin)*(B-T)),!,
     Xint = ((R - L)*(XMax + XMin) - (2*XW - B + T)*(YMax - YMin))div (2*(R - L)),
     Yint = YMin + YW*(YMax - YMin) div (R - L).
convert1(Win,Xint,Yint,XW,YW):-
     limit_coordonates(XMin,XMax,YMin,YMax),
     RCT = win_GetClientRect( Win ),
     RCT = rct(L,T,R,B),
     ((R-L)*(XMax-XMin)) >= ((YMax-YMin)*(B-T)),!,
     Yint = ((2*YW - R + L)*(XMax - XMin) + (B - T)*( YMax + YMin)) div (2*(B - T)),
```

```
        Xint = XMax - XW*(XMax - XMin) div (B - T).


predicates
  win_waypoints_eh : EHANDLER

clauses
  win_waypoints_Create(_Parent):-
        keyword(_,message("Set waypoint XY",_),_),!,
        win_Create(win_waypoints_WinType,win_waypoints_RCT,win_waypoints_Title,
                   win_waypoints_Menu,_Parent,win_waypoints_Flags,win_waypoints_eh,0).

  win_waypoints_Create(_):-!.

%BEGIN Waypoints, e_Create
  win_waypoints_eh(_Win,e_Create(_),0):-!,
        cursor_Set( _Win, idc_zoom ),
        retractall(_,scales),
        assert(limit_coordonates(0,0,0,0)),
        find_limits(1),
        draw_route(_Win),
%BEGIN Waypoints, InitControls, 11:56:52-26.7.1999, Code automatically updated!
%END Waypoints, InitControls
%BEGIN Waypoints, ToolbarCreate, 11:56:52-26.7.1999, Code automatically updated!
%END Waypoints, ToolbarCreate
        !.
%END Waypoints, e_Create
%MARK Waypoints, new events

%BEGIN Waypoints, e_MouseDbl
  win_waypoints_eh(_Win,e_MouseDbl(_PNT,_ShiftCtlAlt,_Button),0):-!,
        Rct = win_GetClientRect( _Win ),
        win_Clear(_Win,Rct,color_White),
        retractall(_,scales),
        assert(limit_coordonates(0,0,0,0)),
        find_limits(1),
        draw_route(_Win),
        !.
%END Waypoints, e_MouseDbl

%BEGIN Waypoints, e_MouseDown
  win_waypoints_eh(_Win,e_MouseDown(PNT,_ShiftCtlAlt,mouse_button_left),0):-!,
        Rct = win_GetClientRect( _Win ),
        win_Clear(_Win,Rct,color_White),
        PNT = pnt(YW,XW),
        convert1(_Win,Xint,Yint,XW,YW),
        retract(limit_coordonates(XMin,XMax,YMin,YMax)),
        XZoom = (XMax - XMin) div 4,
        YZoom = (YMax - YMin) div 4,
        NewXMin = Xint - XZoom,
        NewXMax = Xint + XZoom,
        NewYMin = Yint - YZoom,
        NewYMax = Yint + YZoom,
        assert(limit_coordonates(NewXMin,NewXMax,NewYMin,NewYMax)),
        draw_route(_Win),
        !.

%END Waypoints, e_MouseDown

%BEGIN Waypoints, e_LoseFocus
  win_waypoints_eh(_Win,e_LoseFocus,0):-!,
        win_destroy(_Win),
        !.
```

```
%END Waypoints, e_LoseFocus


%BEGIN Waypoints, e_Update
  win_waypoints_eh(_Win,e_Update(_UpdateRct),0):-!,
       draw_route(_win),
       !.
%END Waypoints, e_Update

%BEGIN Waypoints, e_Size
  win_waypoints_eh(_Win,e_Size(_Width,_Height),0):-!,
       win_Invalidate(_Win),
ifdef use_tbar
       toolbar_Resize(_Win),
enddef
       !.
%END Waypoints, e_Size

%BEGIN Waypoints, e_Menu, Parent window
  win_waypoints_eh(Win,e_Menu(ID,CAS),0):-!,
       PARENT = win_GetParent(Win),
       win_SendEvent(PARENT,e_Menu(ID,CAS)),
       !.
%END Waypoints, e_Menu, Parent window

%END_WIN Waypoints
```

```
TIME_BASED.PRO 7/29/1999

/*****************************************************************************

             Copyright (c) NPS

 Project:  SCRIPT
 FileName: TIME_BASED.PRO
 Purpose: Generation of a Script file
 Written by: Joel Doleac
 Comments: This program is used to create the window where is diplayed the
           commanded trajectory for time based control.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"

%BEGIN_WIN Time Based Flights
/*****************************************************************************
        Creation and event handling for window: Time Based Flights
*****************************************************************************/

constants

%BEGIN Time Based Flights, CreateParms, 15:43:01-25.6.1999, Code automatically updated!
  win_time_based_flights_WinType = w_TopLevel
  win_time_based_flights_Flags =
[wsf_SizeBorder,wsf_TitleBar,wsf_Maximize,wsf_Close,wsf_ClipSiblings]
  win_time_based_flights_RCT = rct(100,80,589,578)
  win_time_based_flights_Menu = no_menu
  win_time_based_flights_Title = "Time Based Flights"
  win_time_based_flights_Help = idh_contents
%END Time Based Flights, CreateParms
domains
  coor = coor(Real,Integer,Integer)

database - time_coordonates
  time_line(Integer,String,coor,coor)

database - scales
  determ limit_coordonates(integer,integer,integer,integer)

predicates
  draw_route(WINDOW,Integer)
  check_arc(WINDOW,Integer,String,coor,coor)
  find_order(Integer,coor,posit)
  find_surge_control(Integer,posit,Integer,Real,coor,coor)
  find_heading(posit,Real)
  find_depth(posit,Real)
  coor_with_angles(Real,coor,coor)
  convert(WINDOW,integer,integer,integer,integer)
  convert1(WINDOW,integer,integer,integer,integer)
  find_limits(posit)
  compare_limits(integer,integer,integer,integer,integer,integer,integer)

clauses

  find_order(Nb,Coor,Posit):-
       counter(Num),
       Posit < Num+1,
       keyword(Posit,message("Set flight duration",[r(T)]),_),!,
       find_surge_control(Posit,Posit,Nb,T,Coor,NewCoor),
       NewPosit = Posit+1,
       NewNb = Nb+1,
```

```
      find_order(NewNb,NewCoor,NewPosit).
  find_order(_,_,Posit):-
      counter(Num),
      Posit > Num,!.
  find_order(Nb,Coor,Posit):-!,
      NewPosit = Posit+1,
      find_order(Nb,Coor,NewPosit).

  find_surge_control(Count,Posit,Nb,T,Coor,NewCoor):-
      NewCount = Count - 1,
      NewCount > 0,
      keyword(NewCount, message("Surge control off",_),_),!,
      find_surge_control(1,Posit,Nb,T,Coor,NewCoor).
  find_surge_control(Count,_,Nb,T,Coor,Coor):-
      NewCount = Count - 1,
      NewCount > 0,
      keyword(NewCount, message("Surge control on",_),_),!,
      assert(time_line(Nb,"Surge control",Coor,coor(T,0,0))).
  find_surge_control(Count,Posit,Nb,T,Coor,NewCoor):-
      NewCount = Count - 1,
      NewCount > 0,!,
      find_surge_control(NewCount,Posit,Nb,T,Coor,NewCoor).
  find_surge_control(1,Posit,Nb,T,Coor1,coor(H,X2,Y2)):-
      find_heading(Posit,H),
      find_depth(Posit,Z),
      coor_with_angles(T,Coor1,coor(H,X2,Y2)),
      format(Text,"% (Heading %, Time %, Depth %)",Nb,H,T,Z),
      assert(time_line(Nb,Text,Coor1,coor(H,X2,Y2))).

  find_heading(Posit,H):-
      NewPosit = Posit - 1,
      NewPosit > 0,
      keyword(NewPosit, message("Heading and sway control",[_,r(H)]),_),!.
  find_heading(Posit,H):-
      NewPosit = Posit - 1,
      NewPosit > 0,
      keyword(NewPosit, message("Set flight heading",[r(H)]),_),!.
  find_heading(Posit,H):-
      NewPosit = Posit - 1,
      NewPosit > 0,
      keyword(NewPosit, message("Rotate",[r(H),_,_]),_),!.
  find_heading(Posit,H):-
      NewPosit = Posit - 1,
      NewPosit > 0,!,
      find_heading(NewPosit,H).
  find_heading(1,0.0).

  find_depth(Posit,Z):-
      NewPosit = Posit - 1,
      NewPosit > 0,
      keyword(NewPosit, message("Set flight depth",[r(Z)]),_),!.
  find_depth(Posit,Z):-
      NewPosit = Posit - 1,
      NewPosit > 0,
      keyword(NewPosit, message("Submerge",[r(Z),_,_]),_),!.
  find_depth(Posit,Z):-
      NewPosit = Posit - 1,
      NewPosit > 0,!,
      find_depth(NewPosit,Z).
  find_depth(1,0.0).

  coor_with_angles(T,coor(H,X1int,Y1int),coor(H,X2int,Y2int)):-!,
      Pi = 3.14159265358979323846264338327950,
```

```
        X2p = T*cos(2*Pi*H/360),
        Y2p = T*sin(2*Pi*H/360),
        Xp = val(integer,X2p),
        Yp = val(integer,Y2p),
        X2int = Xp + X1int,
        Y2int = Yp + Y1int.
  coor_with_angles(T,coor(Hprec,X1int,Y1int),coor(H,X2int,Y2int)):-
        Hprec > H,!,
        Pi = 3.14159265358979323846264338327 95,
        X2p = (T - 2*abs(2*Pi*(H - Hprec)/360))*cos(2*Pi*H/360) + 2*(sin(2*Pi*Hprec/360) -
sin(2*Pi*H/360)),
        Y2p = (T - 2*abs(2*Pi*(H - Hprec)/360))*sin(2*Pi*H/360) + 2*(cos(2*Pi*H/360) -
cos(2*Pi*Hprec/360)),
        Xp = val(integer,X2p),
        Yp = val(integer,Y2p),
        X2int = Xp + X1int,
        Y2int = Yp + Y1int.
  coor_with_angles(T,coor(Hprec,X1int,Y1int),coor(H,X2int,Y2int)):-
        Hprec < H,!,
        Pi = 3.14159265358979323846264338327 95,
        X2p = (T - 2*abs(2*Pi*(H - Hprec)/360))*cos(2*Pi*H/360) - 2*(sin(2*Pi*Hprec/360) -
sin(2*Pi*H/360)),
        Y2p = (T - 2*abs(2*Pi*(H - Hprec)/360))*sin(2*Pi*H/360) - 2*(cos(2*Pi*H/360) -
cos(2*Pi*Hprec/360)),
        Xp = val(integer,X2p),
        Yp = val(integer,Y2p),
        X2int = Xp + X1int,
        Y2int = Yp + Y1int.


  draw_route(Win,Nb):-
        time_line(Nb,Text,Coor1,Coor2),!,
        Pen = pen(1 , ps_Solid, color_Black),
        win_SetPen(Win, Pen),
        win_SetForeColor(Win, color_Black),
        check_arc(Win,Nb,Text,Coor1,Coor2),
        NewNb = Nb+1,
        draw_route(Win,NewNb).
  draw_route(_,_).

  check_arc(Win,Nb,"Surge control",coor(_,X1,Y1),coor(T,_,_)):-!,
        convert(Win,X1,Y1,XW1,YW1),
        format(Text,"% (Surge control: % sec)",Nb,T),
        win_SetForeColor(Win, color_Blue),
        draw_Text(Win, YW1, XW1, Text).
  check_arc(Win,_,Text,coor(H,X1,Y1),coor(H,X2,Y2)):-!,
        convert(Win,X1,Y1,XW1,YW1),
        convert(Win,X2,Y2,XW2,YW2),
        Xtext = (X1 + X2)div 2,
        Ytext = (Y1 + Y2)div 2,
        convert(Win,Xtext,Ytext,XWtext,YWtext),
        draw_Text(Win, YWtext, XWtext, Text),
        draw_Line(Win,pnt(YW1,XW1),pnt(YW2,XW2)),
        T = XW2-4, B = XW2+4,
        L = YW2-4, R = YW2+4,
        Pen = pen(1 , ps_Solid, color_Red),
        win_SetPen(Win, Pen),
        draw_Arc(Win, rct(L,T,R,B), YW2, R, YW2, R).
  check_arc(Win,_,Text,coor(Hprec,X1,Y1),coor(H,X2,Y2)):-
        Hprec > H,!,
        Pi = 3.14159265358979323846264338327 95,
        XendR = X1 + 2*(sin(2*Pi*Hprec/360) - sin(2*Pi*H/360)),
        YendR = Y1 + 2*(cos(2*Pi*H/360) - cos(2*Pi*Hprec/360)),
```

```
      Xend = val(integer,XendR),
      Yend = val(integer,YendR),
      convert(Win,X1,Y1,XW1,YW1),
      convert(Win,Xend,Yend,XWend,YWend),
      convert(Win,X2,Y2,XW2,YW2),
      Tr = 2 + X1 + 2*sin(2*Pi*Hprec/360),
      Rr = 2 + Y1 - 2*cos(2*Pi*Hprec/360),
      Ti = val(integer,Tr),
      Ri = val(integer,Rr),
      Bi = Ti - 4,
      Li = Ri - 4,
      convert(Win,Ti,Ri,T,R),
      convert(Win,Bi,Li,B,L),
      draw_Arc (Win, rct(L,T,R,B), YW1, XW1, YWend, XWend),
      draw_Line(Win,pnt(YWend,XWend),pnt(YW2,XW2)),
      Xtext = (Xend + X2)div 2,
      Ytext = (Yend + Y2)div 2,
      convert(Win,Xtext,Ytext,XWtext,YWtext),
      draw_Text(Win, YWtext, XWtext, Text),
      Ta = XW2-4, Ba = XW2+4,
      La = YW2-4, Ra = YW2+4,
      Pen = pen(1 , ps_Solid, color_Red),
      win_SetPen(Win, Pen),
      draw_Arc(Win, rct(La,Ta,Ra,Ba), YW2, Ra, YW2, Ra).
  check_arc(Win,_,Text,coor(Hprec,X1,Y1),coor(H,X2,Y2)):-
      Hprec < H,!,
      Pi = 3.14159265358979323846264338327 95,
      XendR = X1 - 2*(sin(2*Pi*Hprec/360) - sin(2*Pi*H/360)),
      YendR = Y1 - 2*(cos(2*Pi*H/360) - cos(2*Pi*Hprec/360)),
      Xend = val(integer,XendR),
      Yend = val(integer,YendR),
      convert(Win,X1,Y1,XW1,YW1),
      convert(Win,Xend,Yend,XWend,YWend),
      convert(Win,X2,Y2,XW2,YW2),
      Tr = 2 + X1 - 2*sin(2*Pi*Hprec/360),
      Rr = 2 + Y1 + 2*cos(2*Pi*Hprec/360),
      Ti = val(integer,Tr),
      Ri = val(integer,Rr),
      Bi = Ti - 4,
      Li = Ri - 4,
      convert(Win,Ti,Ri,T,R),
      convert(Win,Bi,Li,B,L),
      draw_Arc (Win, rct(L,T,R,B), YWend, XWend, YW1, XW1),
      draw_Line(Win,pnt(YWend,XWend),pnt(YW2,XW2)),
      Xtext = (Xend + X2)div 2,
      Ytext = (Yend + Y2)div 2,
      convert(Win,Xtext,Ytext,XWtext,YWtext),
      draw_Text(Win, YWtext, XWtext, Text),
      Ta = XW2-4, Ba = XW2+4,
      La = YW2-4, Ra = YW2+4,
      Pen = pen(1 , ps_Solid, color_Red),
      win_SetPen(Win, Pen),
      draw_Arc(Win, rct(La,Ta,Ra,Ba), YW2, Ra, YW2, Ra).


  convert(Win,Xint,Yint,XW,YW):-
      limit_coordonates(XMin,XMax,YMin,YMax),
      RCT = win_GetClientRect( Win ),
      RCT = rct(L,T,R,B),
      ((R-L)*(XMax-XMin)) <= ((YMax-YMin)*(B-T)),!,
      XW = ((XMax - 2*Xint + XMin)*(R - L) + (YMax - YMin)*(B - T))div (2*(YMax -
YMin)),
```

```
        YW = (Yint - YMin)*(R-L) div (YMax-YMin).
    convert(Win,Xint,Yint,XW,YW):-
        limit_coordonates(XMin,XMax,YMin,YMax),
        RCT = win_GetClientRect( Win ),
        RCT = rct(L,T,R,B),
        ((R-L)*(XMax-XMin)) >= ((YMax-YMin)*(B-T)),!,
        YW = ((2*Yint - YMax - YMin)*(B - T) div (XMax - XMin) + R - L) div 2,
        XW = (XMax - Xint)*(B - T) div (XMax - XMin).


    find_limits(Posit):-
        time_line(Posit,_,_,coor(_,X2,Y2)),!,
        retract(limit_coordonates(XMin,XMax,YMin,YMax)),
        compare_limits(X2,Y2,XMin,XMax,YMin,YMax,1),
        NewPosit=Posit+1,
        find_limits(NewPosit).
    find_limits(_):-
        retract(limit_coordonates(XMin,XMax,YMin,YMax)),
        MinX = XMin - 1 -(XMax-XMin)*25 div 100,
        MaxX = XMax + 1 + (XMax-XMin)*25 div 100,
        MinY = YMin - 1 -(YMax-YMin)*25 div 100,
        MaxY = YMax + 1 + (YMax-YMin)*25 div 100,
        assert(limit_coordonates(MinX,MaxX,MinY,MaxY)).


    compare_limits(X,Y,XMin,XMax,YMin,YMax,1):-
        X<XMin,!,
        compare_limits(X,Y,X,XMax,YMin,YMax,2).
    compare_limits(X,Y,XMin,XMax,YMin,YMax,2):-
        X>XMax,!,
        compare_limits(X,Y,XMin,X,YMin,YMax,3).
    compare_limits(X,Y,XMin,XMax,YMin,YMax,3):-
        Y<YMin,!,
        compare_limits(X,Y,XMin,XMax,Y,YMax,4).
    compare_limits(_,Y,XMin,XMax,YMin,YMax,4):-
        Y>YMax,!,
        assert(limit_coordonates(XMin,XMax,YMin,Y)).
    compare_limits(_,_,XMin,XMax,YMin,YMax,4):-!,
        assert(limit_coordonates(XMin,XMax,YMin,YMax)).
    compare_limits(X,Y,XMin,XMax,YMin,YMax,Comp):-!,
        NextComp = Comp + 1,
        compare_limits(X,Y,XMin,XMax,YMin,YMax,NextComp).

    convert1(Win,Xint,Yint,XW,YW):-
        limit_coordonates(XMin,XMax,YMin,YMax),
        RCT = win_GetClientRect( Win ),
        RCT = rct(L,T,R,B),
        ((R-L)*(XMax-XMin)) <= ((YMax-YMin)*(B-T)),!,
        Xint = ((B - 2*XW - T)*(YMax - YMin) + (R - L)*(XMax - XMin))div (2*(R - L)),
        Yint = YMin + YW*(YMax - YMin) div (R - L).
    convert1(Win,Xint,Yint,XW,YW):-
        limit_coordonates(XMin,XMax,YMin,YMax),
        RCT = win_GetClientRect( Win ),
        RCT = rct(L,T,R,B),
        ((R-L)*(XMax-XMin)) >= ((YMax-YMin)*(B-T)),!,
        Yint = ((2*YW - R + L)*(XMax - XMin) div (B-T) + YMax + YMin) div 2,
        Xint = XMax - XW*(XMax - XMin) div (B-T).


predicates

    win_time_based_flights_eh : EHANDLER
```

```
TIME_BASED.PRO 7/29/1999

clauses

  win_time_based_flights_Create(_Parent):-
       keyword(_,message("USE TIME BASED CONTROL",_),_),!,
       win_Create(win_time_based_flights_WinType,win_time_based_flights_RCT,win_time_base
d_flights_Title,

win_time_based_flights_Menu,_Parent,win_time_based_flights_Flags,win_time_based_flights_
eh,0).

  win_time_based_flights_Create(_):-!.

%BEGIN Time Based Flights, e_Create
  win_time_based_flights_eh(_Win,e_Create(_),0):-!,
       cursor_Set( _Win, idc_zoom ),
       retractall(_,time_coordonates),
       retractall(_,scales),
       assert(limit_coordonates(0,0,0,0)),
       find_order(1,coor(0.0,0,0),1),
       find_limits(1),
       draw_route(_Win,1),
%BEGIN Time Based Flights, InitControls, 15:43:01-25.6.1999, Code automatically updated!
%END Time Based Flights, InitControls
%BEGIN Time Based Flights, ToolbarCreate, 15:43:01-25.6.1999, Code automatically
updated!
%END Time Based Flights, ToolbarCreate
       !.
%END Time Based Flights, e_Create
%MARK Time Based Flights, new events

%BEGIN Time Based Flights, e_MouseDbl
  win_time_based_flights_eh(_Win,e_MouseDbl(_PNT,_ShiftCtlAlt,_Button),0):-!,
       Rct = win_GetClientRect( _Win ),
       win_Clear(_Win,Rct,color_White),
       retractall(_,scales),
       assert(limit_coordonates(0,0,0,0)),
       find_limits(1),
       draw_route(_Win,1),
       !.
%END Time Based Flights, e_MouseDbl

%BEGIN Time Based Flights, e_MouseDown
  win_time_based_flights_eh(_Win,e_MouseDown(PNT,_ShiftCtlAlt,mouse_button_left),0):-!,

       Rct = win_GetClientRect( _Win ),
       win_Clear(_Win,Rct,color_White),
       PNT = pnt(YW,XW),
       convert1(_Win,Xint,Yint,XW,YW),
       retract(limit_coordonates(XMin,XMax,YMin,YMax)),
       XZoom = (XMax - XMin) div 4,
       YZoom = (YMax - YMin) div 4,
       NewXMin = Xint - XZoom,
       NewXMax = Xint + XZoom,
       NewYMin = Yint - YZoom,
       NewYMax = Yint + YZoom,
       assert(limit_coordonates(NewXMin,NewXMax,NewYMin,NewYMax)),
       draw_route(_Win,1),
       !.

%END Time Based Flights, e_MouseDown

%BEGIN Time Based Flights, e_Update
  win_time_based_flights_eh(_Win,e_Update(_UpdateRct),0):-!,
```

```
       draw_route(_Win,1),
       !.
%END Time Based Flights, e_Update

%BEGIN Time Based Flights, e_LoseFocus
  win_time_based_flights_eh(_Win,e_LoseFocus,0):-!,
       win_destroy(_Win),
       !.
%END Time Based Flights, e_LoseFocus

%BEGIN Time Based Flights, e_Size
  win_time_based_flights_eh(_Win,e_Size(_Width,_Height),0):-!,
       win_Invalidate(_Win),
ifdef use_tbar
       toolbar_Resize(_Win),
enddef
       !.
%END Time Based Flights, e_Size

%BEGIN Time Based Flights, e_Menu, Parent window
  win_time_based_flights_eh(Win,e_Menu(ID,CAS),0):-!,
       PARENT = win_GetParent(Win),
       win_SendEvent(PARENT,e_Menu(ID,CAS)),
       !.
%END Time Based Flights, e_Menu, Parent window

%END_WIN Time Based Flights
```

```
/*****************************************************************************

             Copyright (c) NPS

 Project:  SCRIPT
 FileName: CHECK_ALL_MODEL.PRO
 Purpose: Generation of the Script file
 Written by: Joel Doleac
 Comments:This program is used to check the model. This function is called in
          the file 'script.pro' by the constant 'id_edit_check_the_model'.
          First, it checks if the keywords 'Initialization done' and
          'Shutdown' exist. Then, it takes a look at each keyword in the
          database and check if they are all at a right position.
*****************************************************************************/

include "script.inc"
include "script.con"
include "hlptopic.con"

predicates
  find_Posit_text(Integer,posit,text)
  check_key_exist(posit,text)
  take_all_keyword(posit,posit,posit)
  nondeterm initialization_text(text)
  nondeterm waypoint_control_text(text)
  nondeterm time_based_control_text(text)
  positioning_logic(posit,text,posit,posit)
  compare(posit,posit,posit,posit)
  compare_not_zero(posit,posit,posit)

clauses
 check_right_model:-
       find_Posit_text(0,IPosit,"Initialization done"),
       check_key_exist(IPosit,"Initialization done"),
       find_Posit_text(0,SPosit,"Shutdown"),
       check_key_exist(SPosit,"Shutdown"),
       find_Posit_text(0,WPosit,"USE WAYPOINT CONTROL"),
       find_Posit_text(0,TBPosit,"USE TIME BASED CONTROL"),
       take_all_keyword(0,WPosit,TBPosit).

  check_key_exist(0,Text):-!,
       format(Note,"The keyword '%' doesn't exist.",Text),
       dlg_Error(Note).
  check_key_exist(_,_).

  find_Posit_text(Nb,NewNb,Text):-
       NewNb = Nb+1,
       keyword(NewNb,message(Text,_),_),!.
  find_Posit_text(Num,0,_):-
       counter(Num),!.
  find_Posit_text(Nb,Posit,Text):-!,
       NewNb = Nb+1,
       find_Posit_text(NewNb,Posit,Text).

  take_all_keyword(Posit,WPosit,TBPosit):-
       NewPosit = Posit+1,
       keyword(NewPosit,message(Text,_),_),!,
       positioning_logic(NewPosit,Text,WPosit,TBPosit),
       take_all_keyword(NewPosit,WPosit,TBPosit).
  take_all_keyword(Num,_,_):-
       counter(Num),!.
```

```
  positioning_logic(Posit,"USE WAYPOINT CONTROL",_,TBPosit):-
       TBPosit<>0,!,
       format(Note,"You cannot have a 'USE WAYPOINT CONTROL' (%) section and a
'USE TIME BASED CONTROL' (%) section in the same script.",Posit,TBPosit),
       dlg_Error(Note).
  positioning_logic(Posit,"USE WAYPOINT CONTROL",WPosit,0):-
       Posit<>WPosit,
       WPosit<>0,!,
       format(Note,"You cannot have two times a keyword 'USE WAYPOINT CONTROL'
(% and %).",WPosit,Posit),
       dlg_Error(Note).
  positioning_logic(Posit,"USE TIME BASED CONTROL",0,TBPosit):-
       Posit<>TBposit,
       TBPosit<>0,!,
       format(Note,"You cannot have two times a keyword 'USE TIME BASED CONTROL'
(% and %).",TBPosit,Posit),
       dlg_Error(Note).
  positioning_logic(_,"USE WAYPOINT CONTROL",_,_):-!.
  positioning_logic(_,"USE TIME BASED CONTROL",_,_):-!.

  positioning_logic(Posit,Text,_,_):-
       not(initialization_text(Text)),
       not(waypoint_control_text(Text)),
       not(time_based_control_text(Text)),!,
       format(Note,"The keyword '%' (%) is not a declared keyword.",Text,Posit),
       dlg_Error(Note).

  positioning_logic(Posit,Text,_,_):-
       not(initialization_text(Text)),
       not(waypoint_control_text(Text)),
       not(time_based_control_text(Text)),!,
       format(Note,"The keyword '%' (%) doesn't exist.",Text,Posit),
       dlg_Error(Note).

  positioning_logic(Posit,Text,0,0):-
       not(initialization_text(Text)),!,
       format(Note,"You cannot put the keyword '%' (%) in the initialization
section.",Text,Posit),
       dlg_Error(Note).


  positioning_logic(Posit,Text,WPosit,0):-
       WPosit<>0,
       Posit < WPosit,
       not(initialization_text(Text)),!,
       format(Note,"You cannot put the keyword '%' (%) in the initialization
section.",Text,Posit),
       dlg_Error(Note).
  positioning_logic(Posit,Text,WPosit,0):-
       WPosit<>0,
       Posit > WPosit,
       not(waypoint_control_text(Text)),!,
       format(Note,"You cannot put the keyword '%' (%) in the 'USE WAYPOINT
CONTROL' section.",Text,Posit),
       dlg_Error(Note).


  positioning_logic(Posit,Text,0,TBPosit):-
       TBPosit<>0,
       Posit < TBPosit,
       not(initialization_text(Text)),!,
       format(Note,"You cannot put the keyword '%' (%) in the initialization
section.",Text,Posit),
```

```
      dlg_Error(Note).
  positioning_logic(Posit,Text,0,TBPosit):-
      TBPosit<>0,
      Posit > TBPosit,
      not(time_based_control_text(Text)),!,
      format(Note,"You cannot put the keyword '%' (%) in the 'USE TIME BASED
CONTROL' section.",Text,Posit),
      dlg_Error(Note).

  positioning_logic(Posit,"Set screw speed",_,_):-
      find_Posit_text(0,SetVolt,"Set screw voltage"),
      SetVolt<>0,!,
      format(Note,"You cannot put the keywords 'Set screw speed' (%) and 'Set
screw voltage' (%) in the same script.",Posit,SetVolt),
      dlg_Error(Note).
  positioning_logic(Posit,"Set screw speed from file",_,_):-
      find_Posit_text(0,SetVolt,"Set screw voltage"),
      SetVolt<>0,!,
      format(Note,"You cannot put the keywords 'Set screw speed from file' (%)
and 'Set screw voltage' (%) in the same script.",Posit,SetVolt),
      dlg_Error(Note).

  positioning_logic(Posit,"Start screw speed control",_,_):-
      find_Posit_text(0,SetPosit,"Set screw speed"),
      find_Posit_text(0,SetFilePosit,"Set screw speed from file"),
      compare(Posit,SetPosit,SetFilePosit,Min),
      Min >= Posit,!,
      format(Note,"The screw speed control cannot start (%) if 'Set screw
speed' doesn't exist.",Posit),
      dlg_Error(Note).
  positioning_logic(Posit,"Start flight heading control",_,_):-
      find_Posit_text(0,SetPosit,"Set flight heading"),
      compare_not_zero(Posit,SetPosit,Min),
      Min >= Posit,!,
      format(Note,"The flight heading control cannot start (%) if 'Set flight
heading' doesn't exist.",Posit),
      dlg_Error(Note).
  positioning_logic(Posit,"Start flight depth control",_,_):-
      find_Posit_text(0,SetPosit,"Set flight depth"),
      compare_not_zero(Posit,SetPosit,Min),
      Min >= Posit,!,
      format(Note,"The flight depth control cannot start (%) if 'Set flight
depth' doesn't exist.",Posit),
      dlg_Error(Note).
  positioning_logic(Posit,"Start screw voltage control",_,_):-
      find_Posit_text(0,SetPosit,"Set screw voltage"),
      compare_not_zero(Posit,SetPosit,Min),
      Min >= Posit,!,
      format(Note,"The screw voltage control cannot start (%) if 'Set screw
voltage' doesn't exist.",Posit),
      dlg_Error(Note).
  positioning_logic(Posit,"Start fixed plane control",_,_):-
      find_Posit_text(0,SetPosit,"Set fixed plane angles"),
      compare_not_zero(Posit,SetPosit,Min),
      Min >= Posit,!,
      format(Note,"The fixed plane control cannot start (%) if 'Set fixed plane
angles' doesn't exist.",Posit),
      dlg_Error(Note).

  positioning_logic(Posit,"Initialization done",WPosit,0):-
      Posit<>WPosit-1,!,
```

```
      format(Note,"The keyword 'Initialization done' (%) has to be the end of
the initialization section. It has to be placed just before 'USE WAYPOINT
CONTROL' (%).",Posit,WPosit),
      dlg_Error(Note).
  positioning_logic(Posit,"Initialization done",0,TBPosit):-
      Posit<>TBPosit-1,!,
      format(Note,"The keyword 'Initialization done' (%) has to be the end of
the initialization section. It has to be placed just before 'USE TIME BASED
CONTROL' (%)",Posit,TBPosit),
      dlg_Error(Note).

  positioning_logic(Posit,"Shutdown",_,_):-
      not(counter(Posit)),!,
      format(Note,"The keyword 'Shutdown' (%) has to be at the end of the
script.",Posit),
      dlg_Error(Note).

  positioning_logic(_,_,_,_).

  compare(_,SetPosit,SetPosFile,SetPosit):-
      SetPosit<>0,
      SetPosFile<>0,
      SetPosit < SetPosFile,!.
  compare(_,SetPosit,0,SetPosit):-
      SetPosit <> 0,!.
  compare(_,SetPosit,SetPosFile,SetPosFile):-
      SetPosit<>0,
      SetPosFile<>0,
      SetPosFile < SetPosit,!.
  compare(_,0,SetPosFile,SetPosFile):-
      SetPosFile <> 0,!.
  compare(Posit,0,0,Posit).

  compare_not_zero(Posit,0,Posit):-!.
  compare_not_zero(_,SetPosit,SetPosit):-!.


  initialization_text(Text):-
      Text = "Turn on ADV power";
      Text = "Turn off ADV power";
      Text = "Turn on sonar power";
      Text = "Turn off sonar power";
      Text = "Get flight controller gains";
      Text = "Get motor controller gains";
      Text = "Set max depth";
      Text = "Set min battery voltage";
      Text = "Initialize boards";
      Text = "Turn on prop power";
      Text = "Turn off prop power";
      Text = "Zero gyros and depth cell";
      Text = "Zero depth cell";
      Text = "Start depth filter";
      Text = "Ignore leak check";
      Text = "Ignore voltage check";
      Text = "Wait";
      Text = "Initialization done";
      Text = "Shutdown".

  waypoint_control_text(Text):-
      Text = "Set screw speed";
      Text = "Set screw speed from file";
      Text = "Start screw speed control";
      Text = "Stop screw speed control";
```

```
      Text = "Start depth error filter";
      Text = "Start heading error filter";
      Text = "Surface";
      Text = "Set screw voltage";
      Text = "Start screw voltage control";
      Text = "Set waypoint XY";
      Text = "Set waypoint GPS";
      Text = "Shutdown".

time_based_control_text(Text):-
      Text = "Set screw speed";
      Text = "Set screw speed from file";
      Text = "Start screw speed control";
      Text = "Stop screw speed control";
      Text = "Set flight heading";
      Text = "Start flight heading control";
      Text = "Stop flight heading control";
      Text = "Set flight depth";
      Text = "Start flight depth control";
      Text = "Stop flight depth control";
      Text = "Set flight duration";
      Text = "Start depth error filter";
      Text = "Start heading error filter";
      Text = "Surge control on";
      Text = "Surge control off";
      Text = "Heading and sway control";
      Text = "Submerge";
      Text = "Rotate";
      Text = "Surface";
      Text = "Set screw voltage";
      Text = "Start screw voltage control";
      Text = "Set fixed plane angles";
      Text = "Start fixed plane control";
      Text = "Shutdown".
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 92943-5101

3. Dr. Donald P. Brutzman, Code UW/Br 1
   Undersea Warfare Academic Group
   Naval Postgraduate School
   Monterey, CA 93943-5100

4. Dr. Anthony J. Healey, Code ME/Hy 1
   Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA 93943-5100

5. Dr. David Marco, Code ME/Ma 1
   Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Dr. Tom Curtin 1
   Office of Naval Research (ONR)
   800 North Quincy Street
   Arlington, Virginia 22217

7. CDR Michael J. Holden, USN, Code CS/Hm 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5100

8. ENIT 3
   47, Avenue d'Azereix
   BP 1629
   65016 Tarbes Cedex, France

9. Joël Doléac 1
   17, rue des platanes
   65800 Orleix, France

10. Didier Léandri 1
"Le Bel Air" esc4
av du 4 Septembre
65000 Tarbes, France

11. Caroline Deltheil 1
Résidence Pullman
13 rue de l'Hourtoulane
66000 Perpignan, France

12. Research Office, Code 09 1
Naval Postgraduate School
Monterey, CA 93943-5000